**UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY**

**Escuela de Ciencias Matemáticas y Computacionales**

# TÍTULO:Drone Detection and Antidrone System Using YOLO: An Effective Approach to Airspace Security

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

**Autor:**

Jaime Vicente Astudillo Tobar

**Tutor:**

Iza Paredes Cristhian Rene, PhD.

Urcuquí, Abril 2024

# Autoría

Yo, **Jaime Vicente Astudillo Tobar**, con cédula de identidad 1718388216, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Abril 2024.

Firmado electrónicamente por:
JAIME VICENTE
ASTUDILLO TOBAR

Jaime Vicente Astudillo Tobar
CI:1718388216

# Autorización de publicación

# Dedication

I dedicate this work to my parents, whose unconditional love and steadfast support have lifted me to heights I never imagined reaching. Their generosity and blind faith in my abilities have illuminated my path throughout this academic journey.
To my dear friends, who have been my support network, sharing both the challenges and the joyful moments during this journey. Your encouragement and friendship have been fundamental pillars at every step of the way.
To Cristhian Iza for his dedication and unconditional support during the development of this work. His invaluable guidance, not only in the academic field but also as a mentor and friend, has been fundamental to my personal and professional growth.
This achievement is not only mine but also yours, and I share it with humility and gratitude.

Jaime Vicente Astudillo Tobar

# Acknowledgment

I offer my profound gratitude to Dr. Cristhian Iza, whose expert guidance and insights proved instrumental throughout this academic journey. To my peer and colleague Robert Chachalo, I express my appreciation for our enlightening discussions and his unwavering support. My deepest thanks extend to my family, who have provided a constant source of encouragement and unwavering love. Lastly, I am grateful to all the authors and researchers whose work has laid the foundation for my research, enabling me to contribute to this field.

Jaime Vicente Astudillo Tobar

# Resumen

En un contexto donde los drones son omnipresentes, resulta crucial abordar los desafíos derivados de su incursión no autorizada en espacios aéreos restringidos. Para ello, este proyecto fusiona algoritmos de detección de objetos YOLO (You Only Look Once) con tecnología de interferencia, buscando proporcionar una solución eficaz y adaptable. Una parte fundamental del proyecto se centra en la implementación y evaluación de la detección de drones basada en YOLO, optando específicamente por la variante YOLOv8 debido a su robustez en esta tarea. A través de rigurosos experimentos y entrenamiento, nos proponemos evaluar el desempeño de YOLOv8 en la detección precisa y eficiente de drones no autorizados en el mundo real. Además de la detección, nos ocupamos del aspecto crítico de la neutralización mediante tecnología de interferencia. Exploramos la integración de MDK4, una herramienta de interferencia disponible en Kali Linux, reconocida por su efectividad en interrumpir la comunicación y las señales de control de los drones. Los resultados y conclusiones obtenidos a lo largo del proyecto arrojan luz sobre la viabilidad y eficacia del sistema de detección de drones basado en YOLO, así como sobre la contramedida de interferencia. Las métricas de rendimiento, que incluyen precisión, recuperación y tasas de falsos positivos/negativos, proporcionan una evaluación integral de las capacidades del sistema.

**Palabras Clave**:
Drones, Acceso no autorizado, Red neuronal convolucional, CNN, Detección de objetos, Yolov8

# Abstract

In a context where drones are ubiquitous, it is crucial to address the challenges arising from their unauthorized incursion into restricted airspace. To this end, this project merges YOLO (You Only Look Once) object detection algorithms with interference technology, aiming to provide an effective and adaptable solution. A key focus of the project lies in implementing and evaluating drone detection based on YOLO, specifically opting for the YOLOv8 variant due to its robustness in this task. Through rigorous experiments and training, we aim to assess the performance of YOLOv8 in accurately and efficiently detecting unauthorized drones in real-world scenarios. In addition to detection, we address the critical aspect of neutralization through interference technology. We explore the integration of MDK4, an interference tool available in Kali Linux, known for its effectiveness in disrupting drone communication and control signals. The results and conclusions obtained throughout the project shed light on the feasibility and effectiveness of the YOLO-based drone detection system, as well as the interference countermeasure. Performance metrics, including precision, recall, and false positive/negative rates, provide a comprehensive evaluation of the system's capabilities.

**Keywords**:
Drones, Unauthorized access, Convolutional neural network, CNN, Object detection, Yolov8.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The proliferation of unmanned aerial vehicles (UAVs), colloquially known as drones, across civilian and commercial domains has precipitated significant challenges for airspace security [1]. Drones, characterized by their accessibility and versatility, have engendered apprehensions pertaining to unauthorized surveillance, airspace violation, and potential malevolent activities, thereby necessitating the development of advanced countermeasures. A comprehensive understanding of these challenges is crucial for effective mitigation strategies. For instance, as depicted in Figure 1.1, the applications of drones span a spectrum ranging from recreational use, such as aerial photography and hobby flying, to agricultural practices like crop monitoring and spraying, and further extending to commercial ventures like package delivery and infrastructure inspection. Additionally, drones have unfortunately been utilized in illicit activities, including smuggling contraband, conducting unauthorized surveillance, and facilitating criminal operations such as the transportation of explosives and bombs [2]. Acknowledging the multifaceted roles of drones underscores the urgency for proactive measures to safeguard airspace integrity and public safety.

(a) Recreational usages

(b) Agriculture

(c) Comercial

(d) Bomb-carrying drones

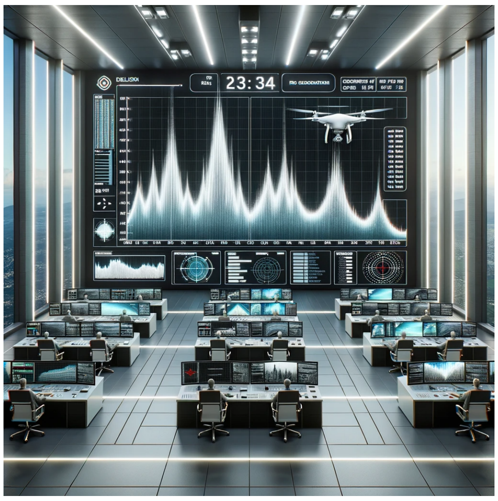
Figure 1.1: Drone applications [3].

Despite their significant benefits, UAVs present a complex regulatory landscape due to potential misuse [4]. Malicious actors experiment with drones across diverse applications, with varying success. In 2023, Ecuadorian police safely detonated an explosives-laden drone within a prison complex in Guayaquil, highlighting security risks [5]. Researchers at the New Mexico Institute of Mining and Technology utilized taxidermied birds as drone platforms [6]. Drones are also widely used in conflicts like the Russia-Ukraine war, and incidents such as airport traffic disruption and privacy intrusion. Hence, there is a pressing demand for the development of effective UAV detection and neutralization techniques [7].

(a) Audio surveillance

(b) Video surveillance

(c) Radar detection



(d) RF analyzer

Figure 1.2: Drone monitoring techniques [3].

A variety of solutions have emerged to address these challenges, including detection and tracking technologies utilizing acoustics, video, radar, and radio frequency, as illustrated in Figure 1.2. Acoustic techniques capture UAV sound signatures effectively, but ambient noise can compromise their effectiveness [8]. Video-based monitoring, augmented by computer vision algorithms, enables real-time identification; however, it struggles under adverse visibility conditions [9]. Radar systems offer superior long-range detection and are robust to weather variations, although detecting smaller drones remains an ongoing challenge [10]. Radio frequency detection excels in early UAV identification, yet it faces limitations with drones using unconventional RF bands or evasion methods [11].

(a) Eagles to hunt drones

(b) Anti-UAV weapon

(c) Digital RF jammer

(d) Electromagnetic pulse

Figure 1.3: Drone neutralisation techniques [3].

Concerning neutralization strategies, a diverse spectrum of techniques exists. These techniques include physical methods such as employing birds of prey, such as eagles, for drone hunting, which have proven effective in capturing and disabling these unmanned aerial vehicles (UAVs) in controlled environments [12]. Additionally, specific anti-UAV weapons have been developed to accurately and safely shoot down drones using a variety of methods such as deployable nets, guided projectiles, and directed laser systems [13]. Furthermore, digital radio frequency (RF) jamming systems have also been used to neutralize drones by blocking their control and navigation signals, hindering their ability to operate effectively [14]. Finally, the use of electromagnetic pulse (EMP) has emerged as a potential technique to disable drones by interfering with their electronic systems, although its effectiveness and feasibility in real-world environments are still being investigated [15]. Figure 1.3 illustrates several of these neutralization techniques.

## 1.1 Problem statement

The escalating prevalence of UAVs, commonly referred to as drones, in both civilian and commercial domains, has raised profound concerns in the realm of airspace security. These concerns stem from the rapid proliferation of drones, which have ushered in a new era of technological capabilities and applications. While drones offer numerous benefits, including surveillance, package delivery, and agricultural monitoring, they have also introduced a host of challenges. One of the primary challenges is the potential misuse of drones for unauthorized activities, such as intrusions into restricted airspace, smuggling, espionage, and even acts of terrorism. These incidents have underscored the pressing need for a robust and multifaceted approach to drone detection and neutralization. This thesis takes on the multifaceted challenge of devising a comprehensive Drone Detection and Antidrone System using YOLO (You Only Look Once) technology, specifically implementing the YOLOv8 variant [16, 17]. YOLO is a state-of-the-art object detection algorithm known for its accuracy and speed in real-time object recognition. By harnessing the power of YOLO, we aim to achieve precise and swift detection of drones within the airspace. Additionally, our system incorporates an antidrone jamming mechanism, which is a pivotal component in neutralizing unauthorized drones. This mechanism disrupts the communication link between the drone and its operator, rendering the drone inoperable and ensuring immediate airspace security. The ultimate challenge of this research is to develop a prototype system that not only excels in drone detection and neutralization but is also well-suited for practical application within the private security sector. This system will be designed to safeguard critical infrastructure, secure public events, and protect against potential threats in various environments.

By exploring and integrating advanced technologies, such as YOLO-based detection and antidrone jamming, we aim to provide an effective and efficient solution to the growing concerns associated with the use of drones in both legitimate and illicit contexts. The development of such a system holds the potential to significantly enhance airspace security in an era where drones are increasingly prevalent and diverse in their applications.

## 1.2 Objectives

### 1.2.1 General Objective

Design and implement system for the detection and neutralization of unauthorized drones.

### 1.2.2 Specific Objectives

1. Conduct a comprehensive review of the literature related to the detection and neutralization of unauthorized drones.

2. Evaluate the performance and effectiveness of the YOLO algorithm as part of the object detection architecture, including its ability for accurate and real-time detection of drones.

3. Assess the effectiveness of detection and the rate of false positives/negatives.

4. Evaluate a neutralization system capable of deactivating unauthorized drones in a controlled environment to ensure its effectiveness and safety.

5. Develop an effective integration interface between the drone detection system and the neutralization system.

6. Verify the integrated system in a controlled setting to guarantee its effectiveness and safety.

# Chapter 2

# Theoretical Framework

The chapter starts by explaining how drones have evolved over time. Then, it discusses in detail the different types, classifications, and uses of drones. Additionally, it introduces the application of Convolutional Neural Networks for object detection, emphasizing the YOLO algorithm. The next section describes how drones use wireless communication links. Finally, the chapter ends by discussing radio interference, which is commonly known as Radio Jamming.

## 2.1 Evolution of Drones

Drones have undergone a remarkable evolution since their inception. Initially conceived for military reconnaissance purposes, they have evolved into versatile tools with applications across various sectors, including commercial, recreational, and scientific domains. The evolution of drones can be traced through significant technological advancements, from their rudimentary beginnings to the sophisticated machines we see today. These advancements are detailed in various articles, including those by [18], [19], [20], [21], [22], [23], [24], [25], [26]. For a detailed overview of this evolution, please refer to Table 2.1.

| **Year** | **Development** | **Description** |
|---|---|---|
| 1849 | Military Balloon | First recorded use of a UAV for military purposes, using a balloon carrier . |
| 1916-1917 | Aerial Target | The development of the first UAVs, primarily used as flying targets for military training. |
| 1939-1945 | Radioplane OQ-2 | Mass-produced during World War II, these were used for anti-aircraft practice. |
| 1960s-1970s | Surveillance Drones | Drones like the AQM-34 Ryan Firebee were used for reconnaissance during the Vietnam War. |
| 1980s | Pioneer UAV | Introduced for naval operations, providing real-time surveillance and damage assessment. |
| 2000s | Modern Military Drones | The MQ-1 Predator and MQ-9 Reaper drones were deployed for surveillance and targeted strikes. |
| 2010s | Consumer Drones | The rise of companies like DJI popularized drones for photography, videography, and recreation. |
| Late 2010s | Commercial Applications | Drones began being used for agriculture, delivery, and environmental monitoring. |
| 2020s | Advanced Autonomy | Integration of AI and machine learning, enabling drones to perform complex tasks autonomously. |

Table 2.1: Evolution of Drones

The technological advancements in drones have been driven by the miniaturization of components, improvements in battery technology, and the integration of advanced sensors and cameras. As drones became more accessible to the general public in the 2010s, their applications expanded beyond military use, leading to innovations in various sectors, from agriculture and real estate to entertainment and logistics [27]. The continuous research in AI and machine learning is further pushing the boundaries of what drones can achieve, making them an indispensable tool in modern society [28].

## 2.2 Types, Classifications, and Applications of Drones

Drones have undergone a remarkable transformation since their inception, initially designed for military purposes but now finding diverse applications across various sectors, from agriculture to entertainment. This theoretical framework delves into the types, classifications, and extensive applications of drones in contemporary society. Table 2.2 presents the classification of unmanned aerial vehicles (UAVs) according to their size, role, operating altitude, and mission radius [29]. The classification categorizes UAVs into three main classes: Class I, Class II, and Class III, each with specific subcategories and examples. Class I includes MICRO, MINI, and LIGHT UAVs, categorized by weight and operating parameters. Class II encompasses TACTICAL UAVs with varying capabilities and mission radii. Class III consists of MALE and HALE UAVs, distinguished by altitude and strategic mission capabilities. This classification system aids in understanding UAV capabilities and selecting suitable platforms for specific operational needs.

| **CLASS** | **Category** | **Role** | **Operating Altitude** | **Mission Radius** | **Example** |
|---|---|---|---|---|---|
| CLASS I ($\leq$ 50 Kg) | MICRO ($<$ 2Kg) | Tactical (Section) | Up to 60 m | 5Km (LOS) | Hummingbird |
| | MINI (2-20 Kg) | Tactical (Company) | Up to 304 m | 25Km (LOS) | Raven |
| | LIGHT ($>$ 20 Kg) | Tactical (Battalion) | Up to 365 m | 50Km (LOS) | Scan Eagle |
| CLASS II ($\leq$ 600 Kg) | TACTICAL | Tactical (Brigade) | Up to 3000 m | 200Km (LOS) | Shadow |
| CLASS III ($>$ 600 Kg) | MALE (Medium Altitude, Long Endurance) | Operational | Up to 13.7 Km | Unlimited (BLOS) | Predator B |
| | HALE (High Altitude, Long Endurance) | Strategic | Up to 19.8 Km | | Global Hawk |
| | Combat | | | | |

Table 2.2: Classification of UAVs.

Other kind of classification can be based on their usage, control type, or physical form. In the following, we are going to detail this taxonomy.

### Usage

- **Military drones**: Historically, drones were primarily developed for military purposes. They are equipped with advanced surveillance systems and, in some cases, weapon systems for targeted strikes. Their roles include reconnaissance, surveillance, and active combat missions [30].
- **Civil drones**: These are non-military drones, further categorized into:
  - **Commercial drones**: Employed in sectors like real estate, agriculture, and entertainment. They offer services ranging from aerial photography to crop monitoring [31].
  - **Hobby drones**: Designed for recreational purposes, these drones are popular among enthusiasts for photography and racing [32].
  - **Governmental drones**: Used by state agencies for purposes like firefighting, search and rescue operations, and traffic monitoring [33].

### Control type

- **Autonomous**: These drones operate without human intervention, relying on integrated systems and sensors. They can make decisions based on the data they gather, making them ideal for tasks that require precision and consistency [34].
- **Monitored**: Requires human oversight. While the drone directs its flight plan, a technician oversees its operations and can intervene if necessary [35].
- **Supervised**: These drones are piloted by an operator but can perform some tasks autonomously. They offer a balance between human control and automation [36].
- **Preprogrammed**: Operates based on a predetermined flight plan. They are ideal for repetitive tasks over the same area, like agricultural surveys [37].
- **Remote controlled (R/C)**: Directly piloted by a technician using a console. These drones are popular for recreational purposes and drone racing [38].

### By Physical form

- **Multicopters**: These drones have multiple rotors, providing them with stability and maneuverability. They are ideal for tasks that require hovering, like aerial photography [39].
- **Helicopters**: Resembling conventional helicopters, these drones offer high load capacity and autonomy. They are often used in specialized applications like heavy cargo transport [40].

- **Fixed-Wing**: These drones are similar to traditional airplanes. They are efficient and can cover large distances, making them suitable for tasks like mapping and surveillance over large areas [41].

### 2.2.1 Applications of drones

The adaptability of drones has led to their widespread adoption across various sectors:

- **Agriculture**: Drones have revolutionized modern farming. They assist in precision agriculture, enabling farmers to monitor crop health, assess irrigation needs, and manage pests. This not only increases yield but also conserves resources [42].
- **Real estate and construction**: Drones offer a bird's-eye view of properties, aiding in property showcases and construction site monitoring. They provide valuable insights into construction progress and site safety [43].
- **Delivery services**: Drones promise faster delivery times, especially beneficial in remote areas or congested urban settings. Companies are exploring drone deliveries for goods, ranging from e-commerce packages to medical supplies [44].
- **Environmental monitoring**: Drones play a pivotal role in monitoring environmental changes. They assist in tracking deforestation, glacier movements, and wildlife. Their ability to access remote areas provides invaluable data for environmental conservation [45].
- **Entertainment**: The entertainment industry leverages drones for capturing aerial shots in movies and organizing drone racing events. They offer unique perspectives that were once challenging or expensive to achieve [46].
- **Emergency response**: Drones are invaluable in search and rescue operations. They can quickly survey disaster-stricken areas, locate victims, and deliver emergency supplies. Their ability to operate in challenging terrains makes them indispensable in such scenarios [37].
- **Surveillance and security**: Drones are increasingly being used for surveillance purposes, from border patrols to crowd monitoring during large events. They enhance security by providing real-time aerial insights [47].
- **Research and development**: Drones serve as platforms for various research purposes. Scientists use them to study weather patterns, volcanic activities, and even to test new technologies and innovations [48].

Table 2.3 displays various categories of rotary UAVs for civilian applications [29]. These UAVs are currently available in the market, predominantly in the mini and micro categories, both of which have limited ranges and heights.

| **Category** | **Acronym** | **Range (Km)** | **Endurance (hrs)** | **Flight Altitude (m)** | **Max Payload (Kg)** |
|---|---|---|---|---|---|
| Micro $< 250$ gr | Micro | $< 10$ | $< 1$ | 250 | $< 5$ |
| Mini $< 25$ | Mini | $< 10$ | $< 2$ | 150 to 300 | $< 30$ |
| Close Range | CR | 10 to 30 | 2 to 4 | 3000 | 150 |
| Short Range | SR | 30 to 70 | 3 to 6 | 3000 | 200 |
| Medium Range | MR | 70 to 200 | 6 to 10 | 5000 | 1250 |
| Low Altitude | LADP | $> 250$ | 0.5 to 1 | 50 to 9000 | 350 |
| Mid Altitude | MRE | $> 500$ | 10 to 18 | 8000 | 1250 |
| High Endurance Low Altitude | LALE | $> 500$ | $> 24$ | 3000 | $< 30$ |
| High Endurance Mid Altitude | MALE | $> 500$ | 24 to 48 | 14000 | 1500 |
| High Endurance High Altitude | HALE | $> 2000$ | 24 to 48 | 20000 | 12000 |
| Combat | UCAV | 1500 | 2 | 10000 | 10000 |
| Offensive | LETH | 300 | 3 to 4 | 4000 | 250 |
| Decoy | DEC | 500 | 4 | 5000 | 250 |
| Stratospheric | STROTO | $> 2000$ | $> 48$ | 2000 to 30000 | ND |
| Exo-stratospheric | EXO | ND | ND | $> 30000$ | ND |

Table 2.3: Classification of UAVs by categories.

The type of controller used determines the UAV's reach among aircraft. Rotary-winged UAVs, like QR-Quadrotors, are perfect for indoor spaces such as stadiums and concerts, making them a favorite in research and development [49]. The ubiquitous use of drones in a variety of applications has generated a growing need to develop effective methods for their detection and tracking in dynamic and variable environments. In this context, CNNs have emerged as a promising tool to address this challenge [50]. By leveraging the CNNs' ability to efficiently process large volumes of visual data, drone detection systems capable of quickly and accurately identifying and tracking these aircraft in real-time images and videos have been developed.

## 2.3 Utilizing Convolutional Neural Networks in Object Detection

The inherent ability of CNNs to learn complex visual patterns and features makes them ideal for addressing drone detection, which often involves small, moving objects in a variety of environments and conditions. CNNs [50] serve as specialized deep neural networks structured in a grid pattern, predominantly tailored for image processing. The defining feature of these networks lies in the employment of convolution, a mathematical operation, rather than the regular matrix multiplication. CNNs trace their lineage to Yann LeCun [51], who drew insights from Kunihiko Fukushima's Neo-Cognitron model. A convolutional operation typically applies an NxN filter to an image's feature map, where $N$ is typically 3. An illustrative example of convolution is shown in Figure 2.1, showcasing edge detection through the convolution process [52].

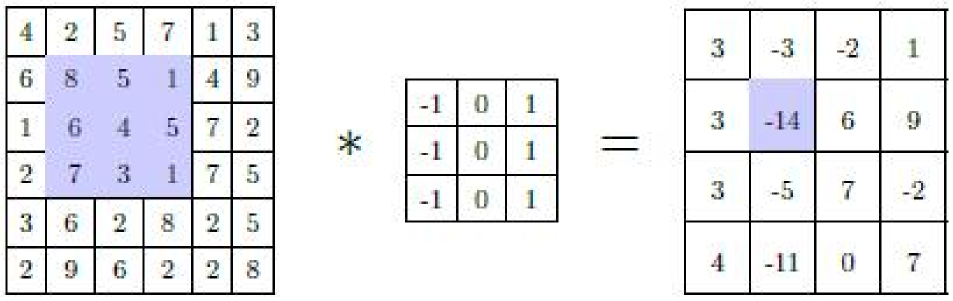


Figure 2.1: Convolution example [52].

## CNN Architecture

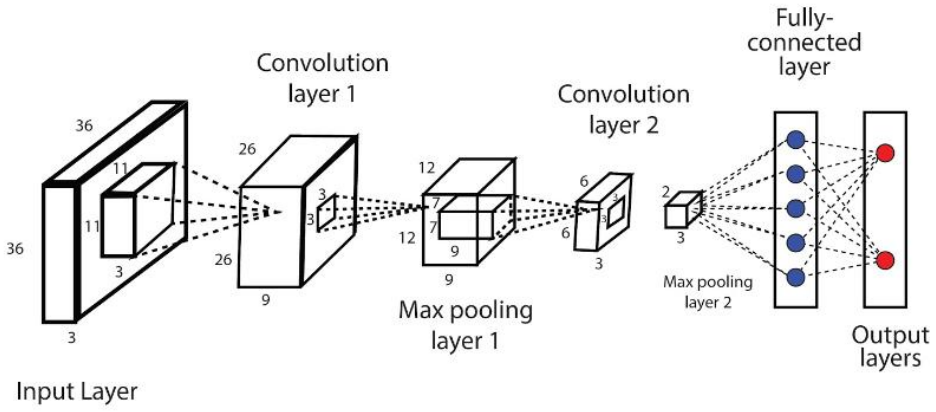


Figure 2.2: Structure of the convolutional neural network [52].

### CNN-Based Detection Models

As shown in Figure 2.2, the application of CNNs to drone detection typically involves adapting the network architecture and training process to effectively identify drones in images or videos captured by drones [52]. Here's how the structure of a CNN might be applied to drone detection:

- **Input layer**: The CNN takes input from images or frames extracted from drone footage.
- **Convolutional layers**: These layers are responsible for extracting features relevant to drones, such as their shape, size, and texture. The convolutional filters convolve across the input images to detect these features.
- **Pooling layers**: Pooling layers help reduce the spatial dimensions of the feature maps generated by the convolutional layers, making the detection process more computationally efficient while retaining important information about potential drone presence.
- **Fully connected layers**: The output from the convolutional and pooling layers is flattened and passed through one or more fully connected layers. These layers help classify whether the detected features correspond to a drone or not.
- **Output layer**: The output layer produces the final predictions, indicating the presence or absence of a drone in the input image or video frame.

CNNs are predominantly harnessed for visual tasks. Numerous models dedicated to object detection have been innovated, some of which are:

- **Faster R-CNN:** This approach integrates a two-phase detector, incorporating a distinct Region Proposal Network (RPN). The RPN processes an image to generate a set of potential object bounding rectangles, which then guide the detection process [53].
- **SSD:** As a single-phase detector, SSD employs varied scaled convolutional bounding box outputs linked to a series of top-tier feature maps [54]. It derives its foundation from the VGG-16 model, conceived by the Visual Geometry Group [55].
- **RetinaNet:** RetinaNet is a singular phase detector amalgamating techniques like anchors and feature pyramids [56, 57]. A groundbreaking addition to this is the Focal Loss, designed to address the disproportion between dominant and recessive features in training one-phase detectors, hence enhancing efficiency.

The utilization of CNNs and convolution operation for drone detection stands as a superior choice over other methods. CNNs, designed specifically for image processing, possess the innate ability to automatically recognize discriminative features from annotated datasets, eliminating the need for manual feature engineering [58]. Furthermore, CNNs demonstrate exceptional scale and orientation invariance, enabling them to identify objects regardless of their size or spatial orientation within images, crucial for effective drone detection in dynamic environments [59]. Additionally, CNNs exhibit robust generalization to new drone images after proper training, adapting well to varying lighting conditions, backgrounds, or drone appearances, ensuring their effectiveness in real-world scenarios like security surveillance or extensive area monitoring [60]. Moreover, CNNs offer notable computational efficiency, particularly in real-time detection in images or videos, which is crucial for practical drone detection systems [61]. In fact, the inherent flexibility and adaptability of CNNs allow seamless adjustments to evolving scenarios or emerging drone variants through additional training data or network architecture modifications, ensuring continued effectiveness against changing threats or challenges [62]. Thus, the integration of CNNs and convolution operation provides a powerful combination of deep learning proficiency, invariance, generalization, computational efficiency, and adaptability, making them highly suitable and efficient for drone detection tasks. In this thesis work, we have chosen the YOLO algorithm [16, 63, 64, 65], as one of the most prominent implementations of CNNs in object detection. YOLO represents a milestone in computer vision by enabling real-time object detection with a single network pass, making it especially relevant in the context of drone detection.

## 2.4 YOLO (You Only Look Once)

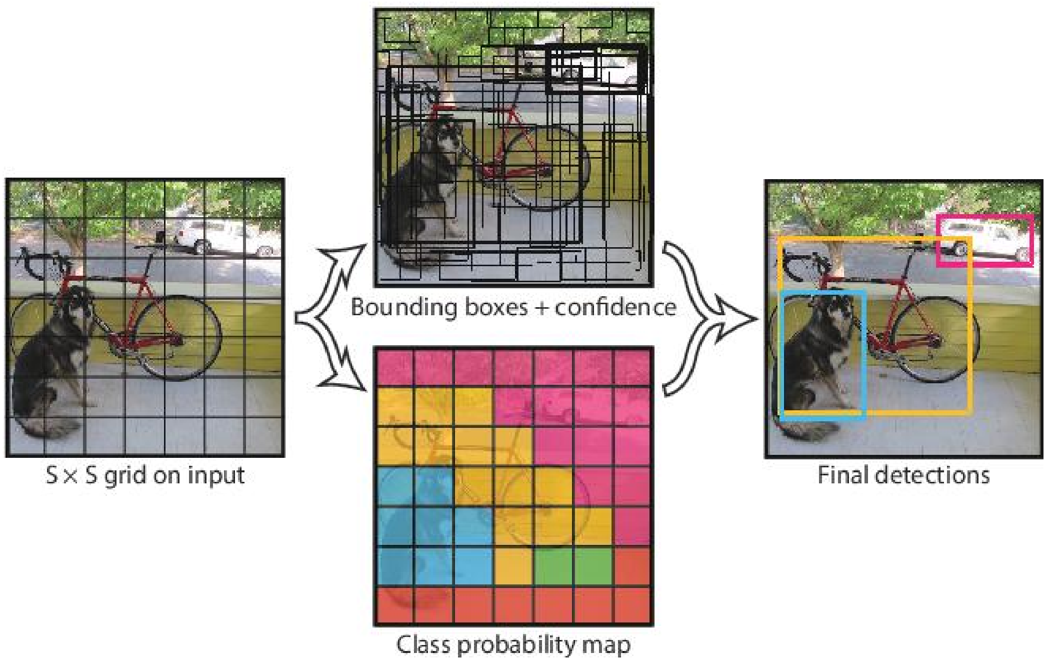


Figure 2.3: The model represents detection as a regression task, dividing the image into an S × S grid. For each grid cell, it predicts B bounding boxes, their confidence scores, and C class probabilities [16].

YOLO revolutionized object detection upon its 2015 introduction by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their work [16]. Unlike traditional methods, YOLO reframes the task as a regression problem rather than classification. This entails segmenting the input image into an SxS grid and employing a single CNN to predict bounding box coordinates and object probabilities for each grid cell. By analyzing each cell individually, YOLO achieves real-time object detection by predicting the presence, position, and confidence of objects in one pass. Figure 2.3 illustrates how YOLO generates class probability maps within grid cells to determine different object categories, resulting in comprehensive object detection across the entire image [16]. YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 frames per second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems, which makes it a great candidate for real-time processing. As depicted in Figure 2.4, the YOLO Architecture [16] operates as follows:

1. Resizes the input image into 448x448 before going through the convolutional network.
2. A 1x1 convolution is first applied to reduce the number of channels, which is then followed by a 3x3 convolution to generate a cuboidal output.
3. The activation function under the hood is ReLU, except for the final layer, which uses a linear activation function.
4. Some additional techniques, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

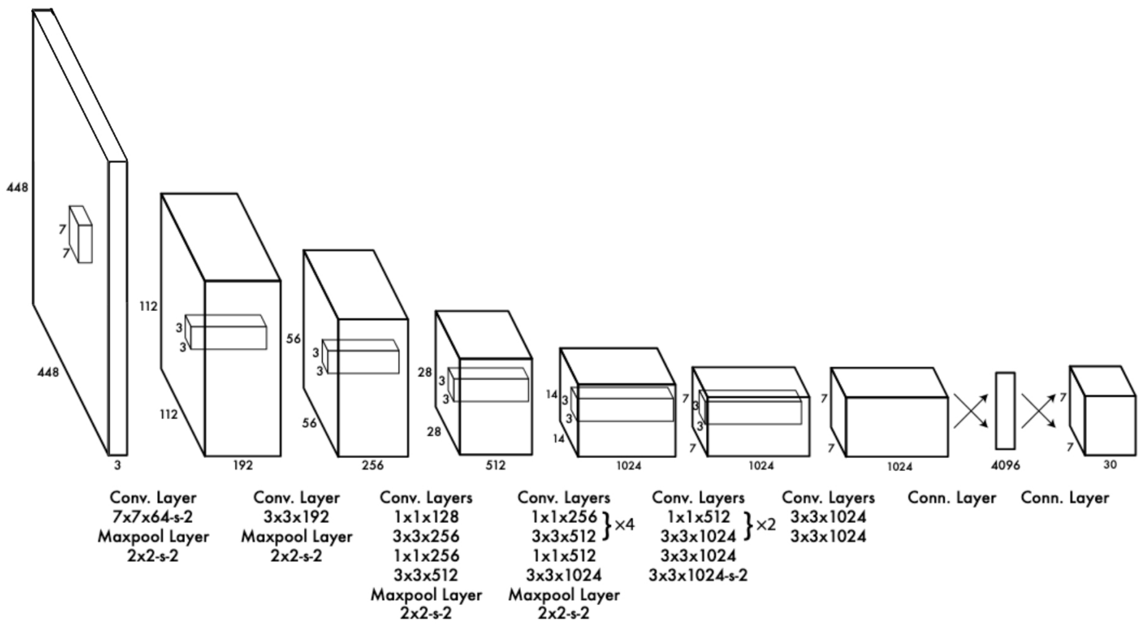

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 2.4: In the YOLO architecture , the network incorporates 24 convolutional layers, followed by 2 fully connected layers. These layers are strategically interspersed with 1 $\times$ 1 convolutional layers to effectively reduce the feature space from the preceding layers. Initially, the convolutional layers undergo pretraining on the ImageNet classification task using images at half the resolution (224 $\times$ 224 input image). Subsequently, the resolution is doubled to enhance detection capabilities [16].

### 2.4.1 Evolution of YOLO Object Detection Model

The development of YOLO has evolved through different versions, from the original YOLO to YOLOv8, each with improvements in terms of accuracy and efficiency. These improvements were based on advances in network architecture, training and optimization techniques. Below we present the different versions of YOLO, from its original version to the latest iterations. We will discover how each version has contributed to the advancement of object detection and how they have been adapted to various applications and scenarios.

- YOLOv2, also known as YOLO9000, was introduced in 2016 and brought foundational improvements over its predecessor [63]. It incorporated concepts such as anchor boxes, inspired by the region proposals of Faster R-CNN. Additionally, multi-scale resizing was introduced to train the network on images of various sizes, enhancing detection across different scales. This version also built upon the Darknet-19 architecture, a 19-layer neural network.

- In 2018, YOLOv3 came forth, offering detection on three scales and the usage of three anchor sizes for each scale [64]. It also expanded its reach to detect up to 80 different classes, making it more versatile for various applications.

- YOLOv4, introduced in 2020, elevated the precision and speed of object detection to a new benchmark [66]. It adopted the CSPDarknet53 as its backbone, incorporated the PANet and SAM block as its neck, and presented the CIOU loss to enhance precision during training. The Mish activation function was also introduced.

- Likewise, YOLOv5, also unveiled in 2020, is not an official continuation of YOLOv4, but has been widely embraced by the community [67]. It offered improvements in speed, a decrease in model size, and facilitated customization and exportation to various platforms.

- YOLO v6, the sixth iteration of the renowned YOLO architecture, has brought forth significant advancements over its predecessor, YOLO v5. One of the key changes in this version is the adoption of a deeper and more complex neural network architecture, which potentially offers improved object detection accuracy. This new architecture is complemented by an enhanced object classification loss function, ensuring better categorization of detected objects. Furthermore, YOLO v6 incorporates a refined offset loss function, which might assist in more accurate bounding box predictions. Another significant enhancement is the improved method for multi-scale object detection, allowing the model to detect objects of various sizes with greater precision [68].

- YOLO v7 continues to push the boundaries of real-time object detection by building upon the foundational improvements of YOLO v6. The neural network architecture has been further refined to achieve even higher detection accuracies. Additionally, YOLO v7 introduces its own refined versions of the object classification loss function and the offset loss function. These modifications are expected to contribute to better object categorization and accurate bounding box predictions. Furthermore, the enhanced method for multi-scale object detection in YOLO v7 ensures that it remains at the forefront of detecting objects across different scales with impeccable accuracy [69].

- Finally, YOLOv8 represents the latest advancement in the YOLO model series, offering robust capabilities for tasks such as object detection, image classification, and instance segmentation.

### 2.4.2 YOLOv8 architecture

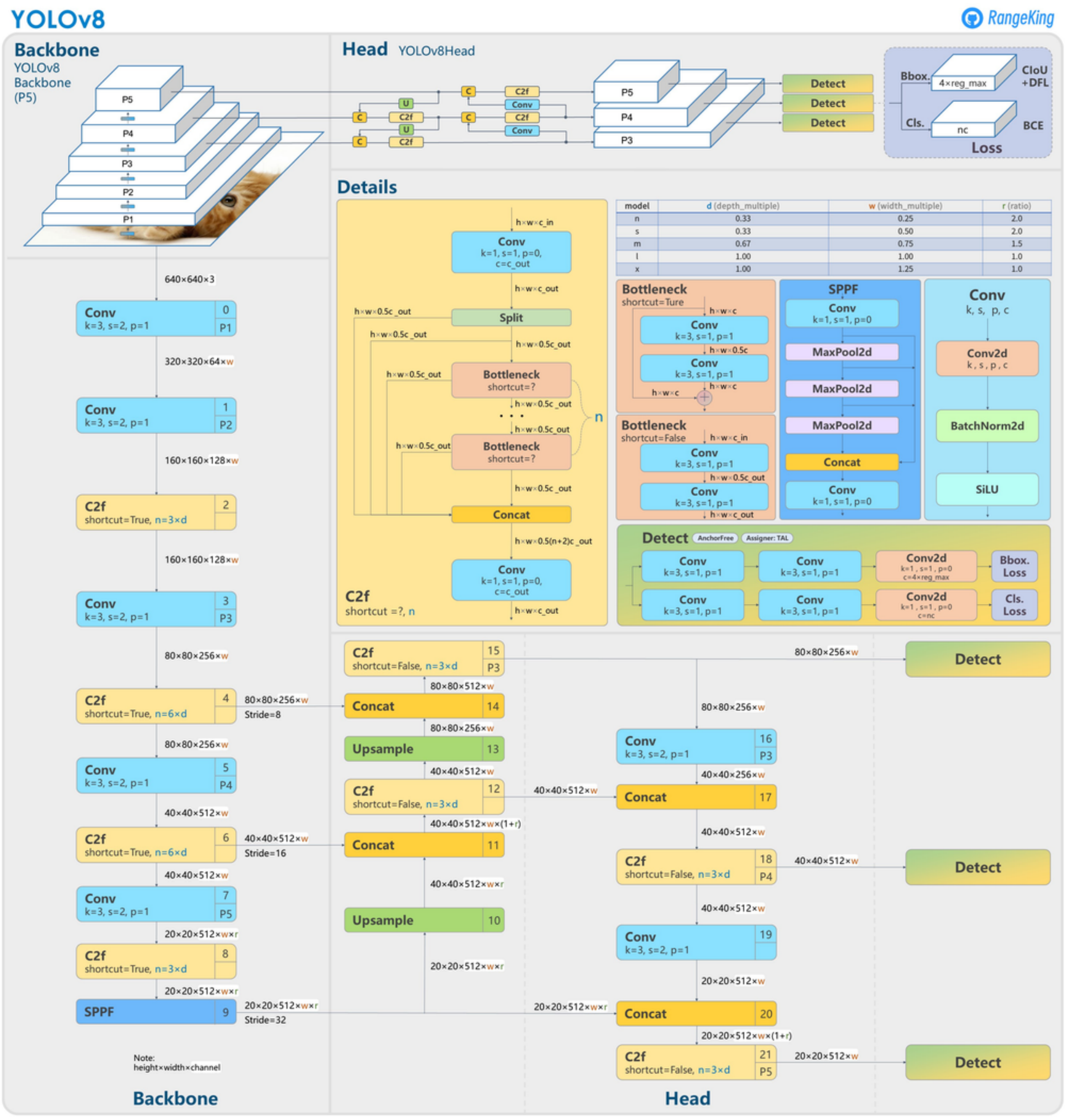
Ultralytics has created YOLOv8, the latest generation of YOLO-based models for Object Detection, which offers exceptional performance [17]. This model is an improvement on previous versions, providing faster and more accurate results. YOLOv8 is a comprehensive platform that allows models to perform Object Detection, Instance Segmentation, and Image Classification. However, the Ultralytics YOLOv8 repository currently lacks some important features, such as a complete set of export features for trained models [17].

YOLOv8 is the first anchor-free model in the YOLO series, meaning that it predicts the center of an image instead of a box. This reduces the number of box predictions, making the non-maximum suppression (NMS) process faster. The C2f is a new convolution layer that concatenates all outputs from the bottleneck, whereas previous versions only used the last bottleneck. In Figure 2.5, the complete architecture is summarized, featuring a close-up of each layer [17]. Additionally, the roboflow team's open-access GitHub repository offers a publicly available YOLOv8 implementation.

### 2.4.3 YOLOv8 features and advantage

Ultralytics have released a completely new repository for YOLO Models. Here are some key features about the new release [17]:

1. User-friendly API (Command Line + Python).
2. Faster and More Accurate.
3. Extensible to all previous versions.
4. New Backbone network.
5. New Anchor-Free head.



| model | d (depth_multiple) | w (width_multiple) | r (ratio) |
|---|---|---|---|
| n | 0.33 | 0.25 | 2.0 |
| s | 0.33 | 0.50 | 2.0 |
| m | 0.67 | 0.75 | 1.5 |
| l | 1.00 | 1.00 | 1.0 |
| x | 1.00 | 1.25 | 1.0 |

Figure 2.5: YOLOv8 model structure [17].

6. New Loss Function.

YOLOv8 is also highly efficient and flexible supporting numerous export formats and the model can run on CPUs and GPUs [17]. The YOLOv8 software package includes several pretrained models that are ready to use. These models include object detection checkpoints that were trained on the COCO detection dataset using images with a resolution of 640. Additionally, this model was evaluated against the previous YOLO versions in different performance metris.

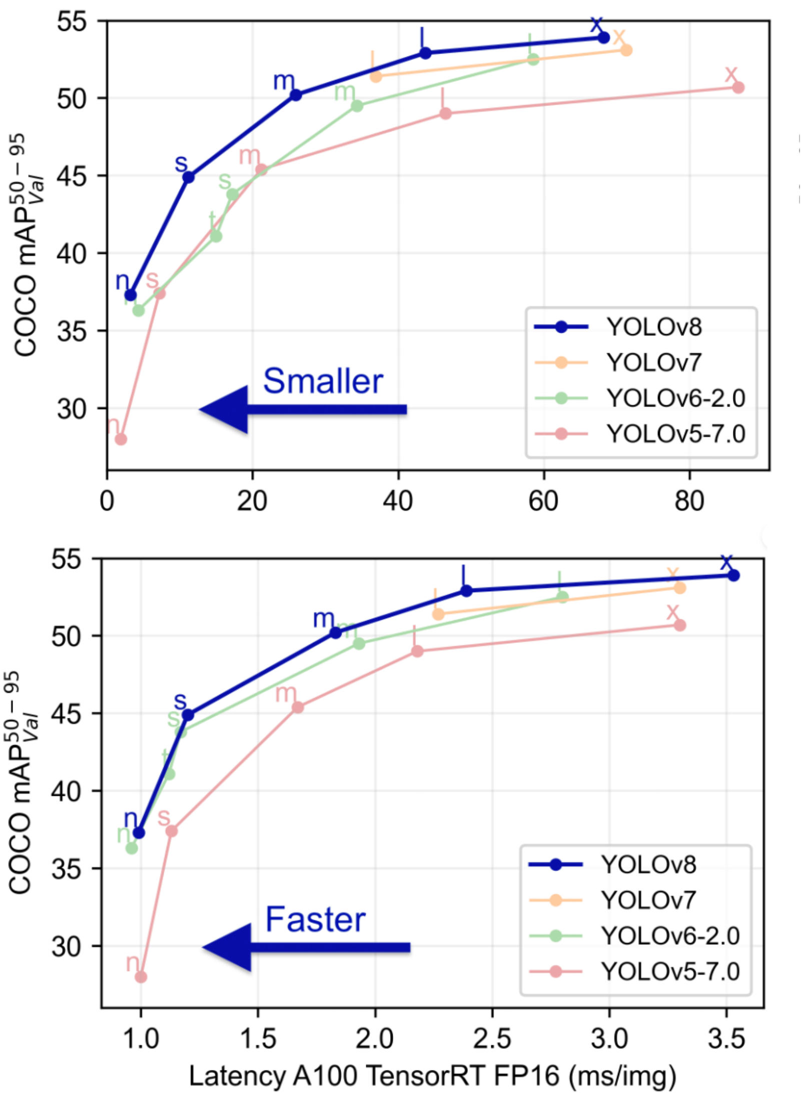


Figure 2.6: Performance evaluation of YOLOv8 with YOLOv5, YOLOv6 and YOLOv7, visualisation made by GitHub user RangeKing [70].

In Figure 2.6, we present an overview of the performance evaluation of YOLOv8 model with YOLOv5, YOLOv6 and YOLOv7 [70]. Its variations, denoted by the suffixes n, s, m, l, and x, represent different model sizes of YOLOv8, each differing in the number of layers, parameters, and overall capacity. Below are succinct descriptions for each size:

1. YOLOv8n: This is the base model characterized by the fewest layers and lowest computational cost. It is suitable for scenarios with limited hardware resources or where speed is of utmost importance.

2. YOLOv8s: A smaller version of YOLOv8 with fewer layers and lower capacity compared to the medium and large models. It strikes a balance between speed and accuracy, making it suitable for real-time applications on devices with moderate computational power.

3. YOLOv8m: A medium-sized model featuring more layers and capacity than YOLOv8s, but less than YOLOv8l and YOLOv8x. It offers a better trade-off between speed and accuracy and is apt for applications where both factors are crucial.

4. YOLOv8l: This is a large model boasting more layers and higher capacity than YOLOv8m, albeit with a higher computational cost. It is suitable for applications prioritizing high accuracy where hardware resources are not a limiting factor.

5. YOLOv8x: The largest and most intricate model within the YOLOv8 series, characterized by the highest number of layers and parameters. While it offers the highest accuracy, it is also the slowest and most computationally expensive, necessitating powerful hardware for efficient execution. It is suitable for applications where achieving high accuracy is imperative, and hardware limitations are not a concern.

### 2.4.4 Performance metrics and initial conditions to train a model

In this section, we will delve into the fundamental metrics utilized to assess the effectiveness of a YOLOv8 model in drone detection, as well as the critical initial conditions required prior to commencing the training phase.

- The mean Average Precision (mAP) is a critical measure in drone detection, providing insight into the effectiveness of detection models. Precision assesses the accuracy of identified drones among all detections, indicating the model's ability to minimize false alarms. It essentially tells us how precise the model is in correctly spotting drones [71]. mAP takes into account precision across various drone categories, offering a comprehensive evaluation of the model's performance. It calculates the average precision for each type of drone, showing how well the model can identify drones accurately in images or videos. A higher mAP score signifies a more dependable and accurate detection model, essential for robust surveillance and security.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{2.1}$$

Where:

> $N$: The total number of object classes evaluated.
>
> $AP_i$: Average Precision of class $i$.
>
> $\frac{1}{N}$: The normalization factor to calculate the average of the APs.

- Recall, a vital metric in classification models, evaluates the model's ability to correctly identify all actual positives, such as drones, among all truly positive instances in the dataset. In drone detection, high recall indicates the model effectively detects drones without missing any, reducing false negatives. To assess a drone detection model, evaluate its performance on test data or cross-validation. Calculate recall for each class (drones) by dividing true positives ($TP$) by the sum of true positives ($TP$) and false negatives ($FN$) [72]. Reporting recall alongside accuracy, precision, and F1 score offers a comprehensive view of the model's effectiveness in detecting drones, aiding in performance assessment and improvement.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Before initiating the training phase, it is essential to establish critical initial conditions to ensure the efficacy of the process.

- Data Augmentation: It is a technique used in machine learning to increase the size and diversity of training data by generating new samples from existing samples. This can include rotating, cropping, enlarging, and changing the brightness and contrast of images, as well as breaking words in a text [73]. The goal is to improve model performance by training it on larger and more diverse data, which helps overcome limitations in data availability and improve the accuracy of models. It also helps to prevent overfitting, which occurs when a model becomes too specialized to the training data and performs poorly on new data. Some of the Data Augmentation techniques we use include: Blur, CLAHE (Contrast Limited Adaptive Histogram Equalization) and ToGray (Turning color images into black and white)

- Transfer learning: It is a machine learning technique in which a model trained for one task is used as a starting point for training another model for another task. This technique is often used to speed up the model training process and also to improve the accuracy of data-constrained machine learning tasks [74]. Furthermore, this technique allows the model to learn useful features from the original task data and transfer them to the task of interest, which is particularly useful when data for the task of interest is sparse or expensive to run. little Transfer learning has been used successfully in various applications such as natural language processing, computer vision, and abnormality detection in medical diagnosis.

### 2.4.5 Potential Use Cases of YoloV8

YOLOv8 presetns a diverse range of applications across object detection and image classification tasks, exemplified in various domains:

- Autonomous vehicles: In the realm of self-driving cars, YOLOv8 offers real-time object detection capabilities, facilitating the identification and tracking of vehicles, pedestrians, and traffic signals, thereby enhancing navigation and safety systems.
- Retail operations: YOLOv8 finds utility in retail environments for inventory management, shoplifting detection, and customer behavior analysis. By monitoring inventory levels and identifying suspicious activities, it aids in optimizing retail operations and ensuring a secure shopping environment.
- Medical imaging: In the medical field, YOLOv8 is utilized for anomaly detection and disease classification in medical imaging. Its ability to detect and classify abnormalities, such as cancerous tumors and fractures, assists healthcare professionals in diagnosis and treatment planning.
- Agricultural monitoring: YOLOv8 plays a pivotal role in agriculture by monitoring crop growth, identifying diseases, and detecting pests. By providing timely insights into crop health and pest infestations, it enables farmers to implement targeted interventions and optimize crop yield.
- Robotics applications: Within robotics, YOLOv8 aids robots in object recognition and interaction within their environment. By accurately detecting and identifying objects, it enhances the autonomy and efficiency of robotic systems in various applications, including manufacturing and logistics.
- Surveillance systems: Within surveillance systems, YOLOv8's capacity for object detection and tracking in real time is invaluable. It enables the surveillance of objects and individuals, contributing to enhanced security measures and threat detection.

## 2.5 Wireless Communication Links Used by drones

In the era of drone technology, one of the crucial components is wireless communication technology that allows the interaction and control of these unmanned aerial vehicles. Drones communicate with their remote controls through wireless communication technologies that allow the transmission of signals and data between the drone and the operator. The most common technologies used for this communication are the following:

1. **Radio frequency (RF)**: RF communication is the standard way to link drones with their controllers, using radio waves to send signals back and forth. Drone controllers often use specific channels, like 2.4 GHz or 5.8 GHz, to cut down on signal mix-ups. This method picks channels that are less crowded to keep the connection clear and control sharp, which is key for flying drones smoothly and safely. Yet, just choosing these channels doesn't completely dodge interference, as other devices might also be using the same space, which can mess with the signal quality.

2. **Bluetooth**: Some consumer drones use Bluetooth technology to communicate with their remote controls. Although Bluetooth typically has a more limited range compared to RF communication, it is suitable for short-range drones and is widely used in toy drones and entry-level drones.

3. **Wi-Fi**: In some cases, drones use Wi-Fi for communication with their controllers. This may allow for a faster connection and extended range compared to Bluetooth, but still has range limitations compared to RF communication.

4. **Mobile App Control**: Some drones, especially entry-level ones, can be controlled via a mobile app instead of a physical remote control. Communication in these cases is usually done via Wi-Fi or Bluetooth, and the mobile phone acts as the remote control.

5. **Custom radio signals**: Custom communication systems like DJI's Lightbridge for drones incorporate advanced modulation techniques, bandwidth optimization, and interference reduction to enhance real-time data and video transmission. They use technologies like MIMO (Multiple Input, Multiple Output) to boost range and reliability, plus encryption for signal security [75]. These systems dynamically switch between channels and frequency bands to minimize latency and maximize connection quality, providing a superior flight experience with precise control and high-quality video transmission, setting them apart from standard RF communication solutions [76, 77].

Drones use wireless transmission and modulation techniques such as OFDM (Orthogonal Frequency Division Multiplexing) and FHSS (Frequency Hopping Spread Spectrum) to establish and maintain reliable and efficient connections between the drone and its remote control or base station [78, 79, 80]. Below is a description of how drones relate to these techniques:

- OFDM is a modulation technique that splits a high-speed signal into multiple slower orthogonal subcarriers. This technique is used in high-speed data transmission, such as real-time video transmission from the drone to the remote control or base station. OFDM is renowned for its ability to mitigate the effects of selective frequency fading and inter-symbol interference [81]. This Frequency Division Multiplexing (FDM) scheme encodes digital data on closely spaced carrier frequencies. With a signal bandwidth ranging from 5 MHz to 40 MHz, the most common overlay protocols are WiFi 802.11n and 802.11ac [82]. Figure 2.7 illustrates an Orthogonal Frequency Division Multiplexing (OFDM) signal utilized in drone wireless communications within the 2.4 GHz band [83]. The blue waveform in the figure represents the OFDM signal, comprising a series of modulated subcarriers. Each subcarrier carries a portion of the data, collectively forming the transmitted signal. For drones, real-time video transmission is essential for applications such as aerial photography and videography [84]. OFDM makes it possible to transmit high-quality video with a high data rate efficiently and resistant to interference.

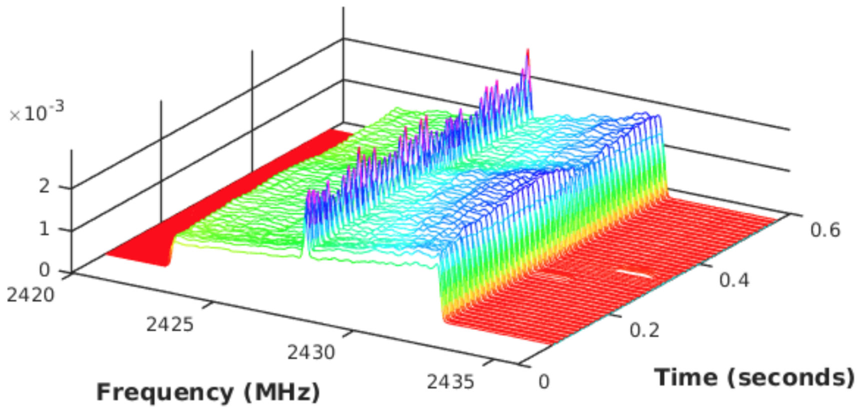


Figure 2.7: The OFDM signal spans over 10MHz. Z-axis represents relative signal strengt [83].

- Frequency Hopping Spread Spectrum (FHSS) is a modulation method frequently employed in drone communications to bolster both reliability and security [85]. In FHSS, both the transmitter and receiver swiftly alternate between numerous narrowband channels within a designated frequency band, such as the commonly utilized

2.4 GHz band for drone operations [86]. Figure 2.8 illustrates this process, where each vertical line signifies a distinct frequency channel within the specified band, and the horizontal axis delineates the sequential order of frequency hops over time [83]. The transmitter undergoes rapid frequency hops according to a predetermined sequence. Each hop occurs momentarily, and the transmitter promptly transitions to the next frequency channel in the predetermined hopping sequence. This cyclical hopping pattern ensures data transmission occurs across various frequencies. The carrier frequencies are spread across uniformly distributed channels throughout a complete frequency band. In this context, it may pertain to the entire 2.4 GHz band (2.4 GHz to 2.5 GHz) or the 5.8 GHz band (5.725 GHz to 5.875 GHz). Usual channels range between 30 and 60, maintaining a 2 MHz gap in between. The bandwidth for a typical channel fluctuates between 1.5 to 2 MHz [87]. FHSS helps ensure robust and reliable communication, even in environments with interference. Drones can change frequencies quickly to maintain a stable connection and minimize the likelihood of data or command loss [86].

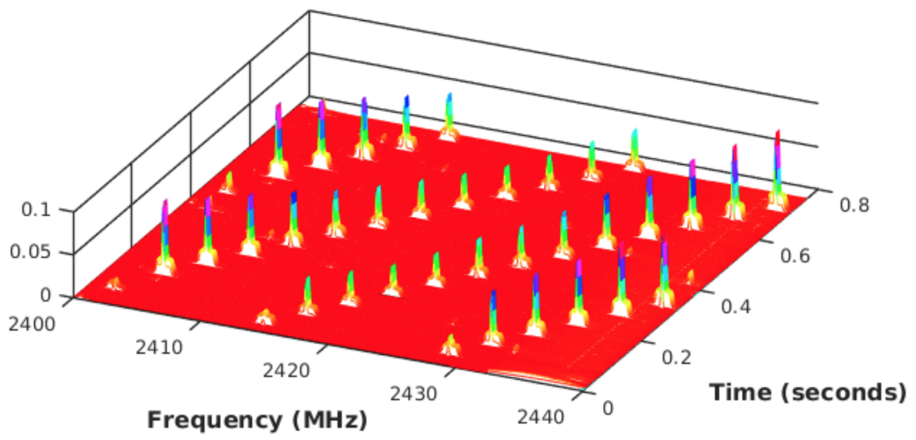


Figure 2.8: The FHSS signal spans over 80MHz with 2MHz wide channels. Z-axis represents relative signal strengt [83].

Both FHSS and OFDM are important in drone communication systems because they enhance the performance, reliability, and security of wireless communication links [78, 79]. By employing these modulation techniques, drones can maintain stable connections with ground stations, control centers, and other drones, even in challenging environments with high levels of electromagnetic interference or signal fading. This ensures smooth and efficient operation of drones, essential for applications such as surveillance, monitoring, mapping, and aerial photography.

## 2.6 Radio Jamming

Radio jamming, or the deliberate interference of drone wireless communications, has emerged as an effective technique to mitigate potential threats and safeguard the integrity of airspace. In this section, we will explore various types of jammers, categorized into three groups based on their influence on the lower layers of the Open Systems Interconnection (OSI) model. Jammers that operate within the context of the Open Systems Interconnection (OSI) model can be categorized based on their influence on the various layers of this model. Specifically, these jammers are divided into three distinct groups, each impacting different aspects of wireless communication [88].

1. **Physical layer**. This category of jammers directly targets the "raw" radio signal transmitted between the sender and receiver. One common technique employed by physical layer jammers is to emit a powerful noise signal, often in various shapes and patterns. The primary objective is to reduce the signal-to-noise ratio (SNR) at the receiver, making it challenging to distinguish the genuine signal from the noise [89, 88]. Within this category, this thesis examines three key types of physical layer jammers: barrage, sweep, and protocol-aware.

    - **Barrage jammer**: This type broadcasts a wideband noise signal, which can span either a single channel or an entire frequency band. Despite its simplicity, implementing a full-band noise jammer can be challenging due to the increased power requirement with greater bandwidth. Moreover, barrage jammers disrupt any form of radio communication, thus causing significant interference [89].
    - **Sweep jammer**: Sweep jammers transmit a narrowband signal that sweeps across the entire frequency band in a continuous motion. The speed at which this motion occurs is referred to as the sweep rate. Compared to barrage jammers, sweep jammers are more straightforward to implement, and under certain conditions, they can behave similarly to barrage jammers [88].
    - **Tone jammer**: Tone jammers divide the noise signal into discrete tones, transmitting them simultaneously or in a sweeping fashion, known as a chirp jammer. While potentially causing less interference with other communications, tone jammers are typically suitable for specific signal types, such as FHSS signals [89].
    - **Protocol-aware jammer**: This jammer mimics the genuine signal as closely as possible, using the same transmitter architecture but transmitting corrupt data. The intention is to increase the likelihood of the receiver picking up the interference signal, as it becomes challenging to filter out the noise signal. However, protocol-aware jammers must be tailored to a particular type of communication, which places additional demands on the detection part of anti-drone systems to correctly identify the target drone's brand, model, and communication type [88].

2. **Data-Link layer jammers**: The data-link layer is responsible for managing data transfer between network nodes. In wireless systems, techniques like Carrier Sense Multiple Access combined with Collision Detection (CSMA/CD) are prevalent. An attack strategy in this layer involves sending brief noise bursts as a wireless device attempts to connect to the network, potentially causing network congestion. However, this approach is less suitable for drone networks where multiple drones do not typically communicate over a single channel [90].

3. **Network layer jammers**: Drones often secure their network layer using encryption and security protocols when operating on WiFi networks. These layers can be targeted in several ways. For example, drones utilizing the 802.11b/g/n protocol can be disconnected from their controller by sending de-authentication commands, forcing the controller to sever the connection with the drone. Continuously sending such commands can prevent the drone from reconnecting to the network [91].

Considering the detailed analysis of radio jamming and its efficacy in mitigating drone threats through the OSI model's lower layers, it is evident that an advanced and nuanced method is necessary to neutralize unauthorized drones effectively.

## 2.7 Wireless Disassociation/Deauthentication Attacks

Disassociation/deauthentication attacks, targeting the 802.11 management frames, exploit weaknesses in the Wi-Fi protocol to forcibly disconnect devices from wireless networks [92, 93]. The attack involves sending forged deauthentication frames to a wireless access point or client device, causing the device to disconnect from the network. The aim of a wireless disassociation attack is to disrupt the normal functioning of a wireless network by inducing client devices to lose their connection [94] . This can pave the way for launching additional attacks, such as man-in-the-middle attacks, or unauthorized acquisition of sensitive information. To execute wireless disassociation attacks, specialized tools or customized scripts are utilized. The attacker must possess access to the wireless network and the ability to transmit deauthentication frames.

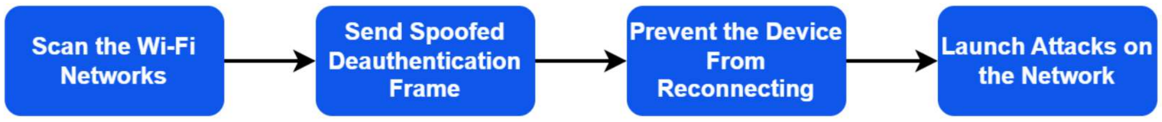

Scan the Wi-Fi Networks
Send Spoofed Deauthentication Frame
Prevent the Device From Reconnecting
Launch Attacks on the Network

Figure 2.9: Wireless Disassociation Attacks

As you can see in Figure 2.9, the wireless disassociation attack commences with the perpetrator engaging in surveillance of the intended wireless network. This involves scanning the WiFi networks through either passive traffic monitoring or active network probing techniques. Upon successfully identifying a target device or access point for disconnection, the attacker proceeds to send spoofed deauthentication frames directed towards the designated device [95]. These frames embody the Media Access Control (MAC) addresses of both the attacker and the target device, in addition to a reason code that ostensibly justifies the disconnection [96]. While this code can assume various values, it's frequently set to 0x0004, signifying deauthentication purportedly due to inactivity. Upon receipt of these forged deauthentication frames, the target device inadvertently perceives them as legitimate directives and consequently severs its connection with the network, preventing the device from reconnecting. At this juncture, the client device may initiate attempts to re-establish network connectivity, while the attacker continues to launch attacks on the network. MDK4 is an exemplary tool in this regard, offering the capability to exploit weaknesses in wireless communication protocols.

### 2.7.1 MDK4

MDK4, also known as Murder Death Kill 4, is a powerful Wi-Fi jamming tool used primarily for testing the security of wireless networks. It operates by sending deauthentication frames to target devices, causing them to disconnect from their network temporarily or indefinitely. Here's how MDK4 works:

- Scanning for networks: MDK4 begins by scanning the surrounding area for available Wi-Fi networks. It collects information about nearby access points, including their SSIDs (network names) and MAC addresses.

- Selecting a target: The user selects a target network to disrupt. This could be their own network for testing purposes or a network they are authorized to test.

- Sending deauthentication frames: MDK4 sends deauthentication frames to the target network's MAC address, pretending to be the access point. These frames instruct the client devices connected to the network to disconnect temporarily.

- Repeating the attack: MDK4 repeats the deauthentication attack at regular intervals, ensuring that any devices attempting to reconnect to the network are immediately disconnected again. This can effectively prevent legitimate users from accessing the network.

- Monitoring the attack: Throughout the process, MDK4 provides feedback to the user, displaying information about the progress of the attack and any devices affected.

Its array of attack modes, particularly the deauthentication and disassociation tactics, align precisely with the vulnerabilities highlighted by network layer jammers. These strategies can disrupt the critical connection between a drone and its controller, ensuring the protection of secure airspace. Furthermore, MDK4's potential to create noise interference across various frequencies complements the strategies within the physical layer, supplementing the functions of barrage and sweep jammers. The tool's protocol-aware jamming feature can be customized to target specific drone communications, which echoes the necessity for precise identification of drone types as discussed earlier. Therefore, the selection of MDK4 for drone neutralization in this thesis is a strategic response to the multifaceted threats posed by unauthorized UAVs. By leveraging MDK4's robust functionalities, this research aims to establish a defense mechanism that is both dynamic and resilient, capable of adapting to evolving drone technologies and the myriad of challenges they present [97, 98, 99].

## 2.8 Security evolution in drone-controller wireless connections

The progression of security measures in wireless communications between drones and their remote controllers, encompassing advancements from legacy protocols to contemporary standards like WPA3, stands as a notable trend. This evolution mirrors the escalating ubiquity of remote control devices for drones [100, 101]. With the introduction of WPA2 in 2004, a substantial leap forward was achieved in contrast to its precursor, WPA, effectively becoming a prerequisite for Wi-Fi Alliance certification from March 2006 until June 2020 [102, 103]. Authentication mechanisms within WPA2 networks pivot on controllers possessing an authentication key, thus underpinning its extensive proliferation, which accounted for a dominant share of drone wireless connections in 2020, comprising 68.75% [104, 105]. A pivotal divergence between WPA2 and WPA3 manifests in the realm of frame management. Unlike WPA2, in WPA3, management frames undergo authentication, effectively curtailing vulnerabilities to spoofing and preempting deauthentication and disassociation attacks [106, 107]. Furthermore, the enforcement of Management Frame Protection (MFP) within WPA3 reinforces security measures, amplifying its robustness against malicious intrusions [108, 96]. Despite these strides, the assimilation of WPA3 remains modest in wireless connections between drones and their controllers, with a mere 1.03% adoption rate as of March 01, 2024, as per data extracted from the Wigle.net database [109]. Moreover, lingering apprehensions persist regarding plausible vulnerabilities in WPA3, inclusive of downgrade attacks, denial-of-service attacks, connection deprivation attacks, bad-token vulnerability and active dictionary attacks [108, 110, 100, 111].

# Chapter 3

# State of the Art

This chapter describes some advancements in object detection using the YOLO algorithm, emphasizing its application in detecting and neutralizing unauthorized drones, and highlights the need for robust security systems in response to the rising drone threats.

## 3.1 You Only Look Once (YOLO)

Object detection in computer vision has been revolutionized by the YOLO algorithm, renowned for its real-time precision [112]. Unlike approaches based on classifiers, YOLO is trained directly on full images and uses a loss function that directly corresponds to detection performance. Numerous studies and incremental improvements have been carried out over the years. Redmon et al. initially introduced the YOLO algorithm in 2016 [112], and since then improvements such as YOLOv3 [113], YOLOv4 [114], YOLOv5 [115], YOLOv6 [116], and YOLOv7 [117] have been proposed. These works have addressed different aspects of the algorithm, such as accuracy, speed, and efficiency. For instance, Bochkovskiy et al. in their work on YOLOv4 achieved an optimal balance between speed and accuracy, establishing a new state-of-the-art in real-time object detection [114]. Furthermore, Hurtik et al. proposed Poly-YOLO, an enhancement of YOLOv3 that achieves more accurate detection and instance segmentation [118]. In the same line, Y. Zhou et al proposed an improved model, "Improved You Only Look Once and None Left" (IYOLO-NL), based on YOLOv5 for real-time facial mask detection, addressing challenges such as occlusion and scale variation [119]. Furthermore, the integration of architectures like CSPNet-Ghost into YOLO models has been shown to significantly improve efficiency in object detection [120]. The creation of specific datasets, like the Face Mask Dataset (FMD), has been vital for training and validating these advanced models [121]. These advancements demonstrate the ongoing evolution of object detection with YOLO, each version striving to surpass its predecessors [122, 123, 112, 113].

### 3.1.1 YOLO Aplication

YOLO has been utilized in various fields such as agriculture, transportation, healthcare, security among others. In the field of agriculture, it has been employed for the detection and tracking of animals in images captured by drones. For instance, in [124],authors used the enhanced YOLOv5s algorithm to count and detect reindeer and sika deer in drone-captured images. The experimental results showcased high precision in detecting these animals, which can be highly beneficial for wildlife monitoring and preventing crop damage. In the transportation field, the YOLO algorithm has been applied for detecting vehicles and pedestrians to enhance road safety. Likewise, in [125], authors used YOLOv1 for detecting missing victims in urban search and rescue scenarios. Through camera utilization, the algorithm was able to identify and locate missing individuals, facilitating rescue operations. In the healthcare sector, the YOLO algorithm has been employed for detecting and classifying cancer cells in histopathological images. For example, in [126], the YOLOv2 algorithm was used to detect cancer cells in breast cancer tissue images. The results demonstrated high precision in cancer cell detection, which can be instrumental for early diagnosis and cancer treatment. In the surveillance and security field, the YOLO algorithm has been used for detecting suspicious or unauthorized objects.for instance, in [127], authors developed a system for detecting and tracking cooperative drones using YOLOv2. This system enabled real-time identification and tracking of drones, which can be crucial for protecting critical infrastructures and preventing illicit activities. This analysis indicates that the YOLO algorithm holds significant potential across a wide range of fields.

## 3.2 Anti-Drone System

The proliferation of drones has drawn attention to the potential risks they pose, particularly when operated by criminal entities [128]. In response, advanced anti-drone systems have been developed, prominently incorporating the YOLO algorithm for its real-time object detection capabilities [129]. Recent research highlighted a technique based on deep neural networks, which uses YOLO to detect interfered echoes, addressing challenges related to interrupted sampling and interference deception [130]. Comparative analysis indicates that YOLO-based systems outperform traditional approaches in terms of accuracy and operational speed [131]. YOLO's real-time operation increases its role as a key component in identifying and countering unauthorized drones [132].

Ensuring early identification of unauthorized drones is critical to mitigating threats, especially in high-risk areas such as airports, prisons, major event venues, stadiums, and private property [133]. When drones employ jamming mechanisms, particularly those based on Digital Radio Frequency Memory (DRFM), the YOLO framework proves effective in distinguishing genuine drones from deceptive targets set by jammers [130]. However, detection is only one part of security; neutralizing these unauthorized drones is of equal importance. Systems such as Orelia Drone-Detector and DroneDetector have been recognized for their effectiveness in this domain [134, 135]. Dedrone, for example, has devised solutions that combine various detection methods to identify and counter drones at varying distances [136]. Proposals have also emerged for comprehensive systems, including signal detection antennas, drone identification modules and jamming components [137]. These systems are designed not only to identify but also to disarm potential aerial threats, ensuring protection in vital regions.

As drone technologies evolve, detection and neutralization methodologies adapt at the same time. However, this evolution faces new challenges. Modern drones are now integrating evasion strategies, including camouflage, which complicates their detection processes [132]. Furthermore, intentional disruptions, particularly through the deployment of jammers by malicious users, further test the resilience of detection systems [138]. Therefore, it becomes imperative that countermeasures advance in tandem with emerging threats. One possible approach is the combination of various detection strategies, encompassing audio, RF, and computer vision-based methods [134, 135]. Ongoing research into artificial intelligence and deep learning promises to refine the precision and effectiveness of these countermeasures [120, 122, 130]. The systems responsible for detecting and neutralizing drones must be reliable and accurate, minimizing false detections and while ensuring that genuine threats are addressed [136, 137].

Despite advancements in drone detection and anti-drone systems, there remains a gap in the literature regarding an integrated system that combines both aspects. The need for a system that can automatically detect unauthorized drones and activate real-time anti-drone measures is evident. Integrating deep learning techniques with advanced jamming systems might be the key to developing more robust and effective solutions in the future.

# Chapter 4

# Methodology

In this chapter, we outline the steps we'll take to build a prototype for drone detection and countermeasures, with a focus on improving airspace security. We'll provide a clear road-map for our research, including the stages of research, design, and implementation, all leading to the creation of a strong and effective system.

## 4.1 Phases of Problem Solving

Our primary concern is the growing challenges posed by drones, which require us to secure airspace against potential misuse. Drones offer many benefits but also raise issues related to safety and security. The central problem we aim to solve is: How can we create a system that reliably detects drones and takes action to neutralize potential threats?

To address this multifaceted challenge, a two-fold solution is required. First, the accurate detection of drones is imperative for threat identification. Traditional radar systems, while effective in certain contexts, may falter when dealing with small, low-flying drones. Advanced detection technologies are needed to swiftly and accurately identify drones in diverse scenarios. Second, an effective countermeasure must be promptly deployed upon drone detection. Countermeasures encompass a range of strategies, including signal jamming, interception, and incapacitation. The synchronization and coordination of detection and countermeasure systems pose a complex challenge.

This thesis focuses on developing a comprehensive Drone Detection and Countermeasure Prototype, harnessing YOLOv8 for detection and jamming technology for countermeasures. The prototype will be designed to:

- **Detect drones:** Utilize YOLOv8 to identify and classify drones accurately, minimizing false positives and false negatives across diverse scenarios.

- **Counter drone threats:** Implement jamming technology as a countermeasure to neutralize detected drones effectively, ensuring public safety and safeguarding critical infrastructure.
- **Coordinate detection and countermeasures:** Establish a seamless coordination mechanism between detection and countermeasure systems, enabling real-time decision-making and threat neutralization.
- **Enhance privacy and safety**: Address privacy concerns by ensuring the prototype operates within legal and ethical boundaries, respecting individual rights and safety.

## 4.2 YOLOv8: Advancing Drone Detection

Drone detection has become a critical challenge in today's world. In this context, the YOLOv8 algorithm has emerged as a cutting-edge tool for drone detection, marking a significant milestone in this field. One of the most notable aspects of YOLOv8 is its ability to detect drones with precision and real-time speed. This is crucial for security as it allows for fast and accurate drone detection, even in challenging situations such as varying lighting conditions or changing speeds. YOLOv8 also provides the capability to identify multiple classes of objects, meaning it can not only detect drones but also other elements present in the environment. This versatility is essential in practical applications where drones may interact with various objects, such as buildings, vehicles, and people. The evolution of YOLOv8 has focused on improving detection accuracy and speed. This has been achieved through the optimization of its architecture and the application of advanced deep learning techniques. Furthermore, YOLOv8 has been trained with a large amount of drone image data, enhancing its ability to recognize a wide variety of drone models and configurations.

Below, we outline the procedures followed to perform drone detection using the YOLO algorithm. This process includes dataset creation, training of various YOLO variants, the necessary code implementation, and concluding performance tests of the algorithm.

### 4.2.1 Dataset

Building a labeled dataset is one of the most crucial steps in training neural networks, as its quality directly impacts the results. For this task, a selection of images needs to be made and accurately labeled. Two methods have been employed to choose the photographs that ultimately compose the dataset. The first method involves searching in public datasets, while the second method entails generating our own images and manually labeling them.

Initially, the plan was to search for drone's eye-view images on the internet, as creating the dataset entirely manually appeared to be a labor-intensive task. Several public datasets were found that compiled a large number of images of this kind. One such dataset is

Roboflow, a project containing aerial detections of various objects, such as cars or people. Initially, time was dedicated to collecting and filtering this dataset for model training, as it featured detections of various objects, and many photos did not contain drones, rendering them useless. Roboflow's dataset not only provided sequences of drone images but also the corresponding annotations for bounding boxes of detections for different YOLO versions. Subsequently, other datasets that met the required criteria were explored by investigating various websites, but they were eventually discarded. Since there was the option to create a new dataset manually, it was finally decided to proceed with this approach. Additionally, another dataset was found that contained labels for each different type of drone (unlike the previous dataset obtained from the website). This dataset was ultimately used for conducting a series of tests.

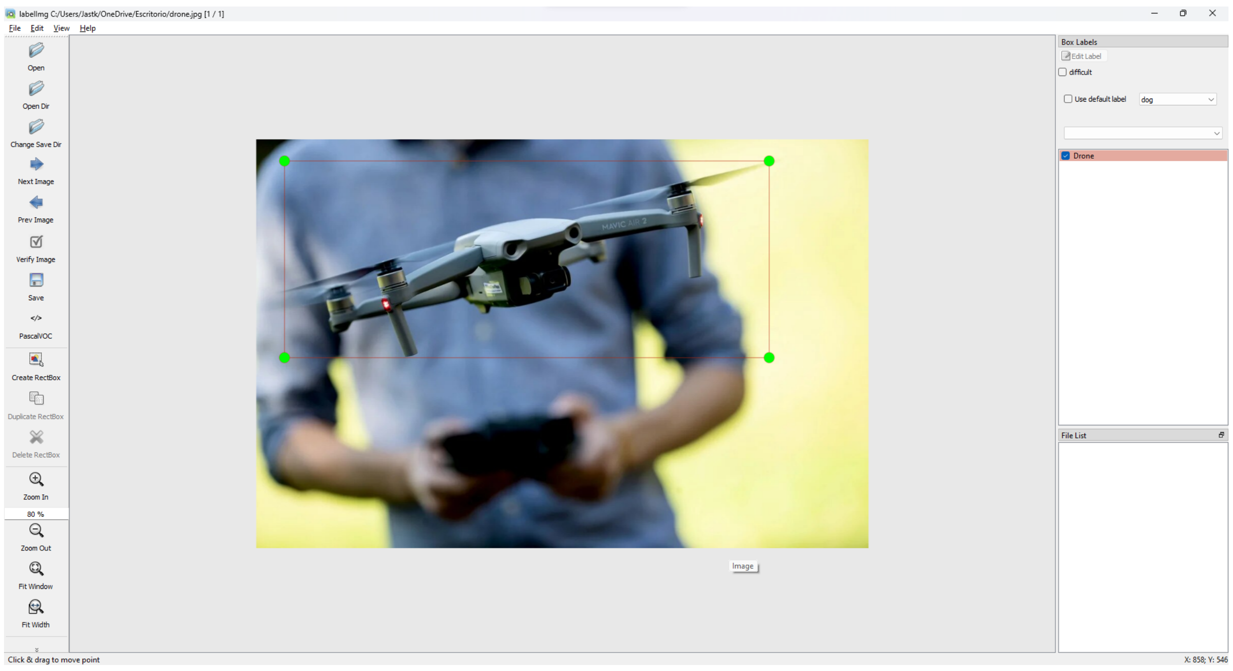


Figure 4.1: LabelImg Tool Interface

To construct a dataset manually, it was necessary to gather a sequence of images. Various types of drones were used in different scenarios, such as roads or pathways. To label the drones in these images, LabelImg was employed. This tool facilitates manual labeling (see Figure 4.1), allowing for the placement of bounding boxes around the drones in the image, the addition of a label, and the saving of this bounding box in the selected format, in this case, YOLO. Consequently, the information is stored in a text document with the same name as the image, beginning with a number representing the label (all belonging to the same class, thus set as 0), followed by four numbers indicating the coordinates of the bounding box. The dataset utilized in this work consists of drone images for the data collection process, comprising a total of 8593 images captured in various environments. This dataset serves the purpose of identifying drones and their in-flight processes, offering utility in storage, management, and detection of drones in motion.

It's important to note that most images in the dataset share a similar background (an open area). While this characteristic can be advantageous for certain applications, it may present a limitation if the model is deployed in a system that detects drones in natural environments, as the model would not have been trained with backgrounds featuring plants, trees, and animals, potentially resulting in reduced performance. However, in scenarios with clear backgrounds, this dataset remains robust.

| **Hardware Details** | **Asus CPU** | **NVIDIA RTX 2060** | **CEDIA HPC CPU** | **CEDIA HPC GPU** | **Raspberry Pi 4** |
|---|---|---|---|---|---|
| Architecture | Zen 2 | Turing | NUMA | Ampere | ARM Cortex-A72 |
| Model | Ryzen 7 4000 series | NVIDIA GeForce RTX 2060 | AMD EPYC 7742 | NVIDIA A100-SXM | Raspberry Pi 4 Model B |
| Number of Cores | 8 cores | 1920 CUDA Cores | 64 cores | 432 tensor cores | 4 cores |
| Clock Speed | 4.5 GHz | 1.365 GHz (Boost Clock: 1.68 GHz) | 3.400 GHz | 1.410 GHz | 1.5 GHz |
| Bus Speed | 4 GT/s | 14 Gbps (GDDR6 Effective) | 16.0 GT/s | 64 GT/s | N/A |
| RAM Capacity | 32 GB | 6 GB (GDDR6) | N/A | 80 GB | 4 GB |
| Operating System | Ubuntu 22.04 | Linux | Ubuntu 18 | Ubuntu 18 | Raspbian |
| CUDA | N/A | 7.5 | N/A | v 11.7 | N/A |
| Pipelines / Threads | 16 threads | 1920 CUDA Cores | 128 threads | 6912 FP32 CUDA cores | N/A |

Table 4.1: Technical specifications of different platforms and hardware components.

### 4.2.2 Hardware description for the training of drone detection models

Table 4.1 summarizes the specifications of various hardware components used in this research project for the training of drone detection models. This hardware set includes a variety of processing platforms, from powerful CPUs and GPUs designed for intensive computing tasks, such as the Asus CPU with Ryzen 7 4000 series and the NVIDIA RTX 2060, to specialized high-performance solutions like the CEDIA HPC CPU with AMD EPYC 7742 and the CEDIA HPC GPU with NVIDIA A100-SXM, complemented by low-cost and energy-efficient devices like the Raspberry Pi 4. Each component has been selected to leverage its unique capabilities at different stages of the training and evaluation process, thus optimizing the performance and efficiency of the drone detection system across a diversity of operational scenarios.

### 4.2.3 Exploring YOLO Algorithm Variants: From YOLOv4 to YOLOv8

of drone detection in different scenarios and conditions. The selection and experimentation with different versions of the YOLO algorithm in development environments like Google Colab and CEDIA (Center for Research and Development in Electronics and Information Technologies) mark a pivotal stage in the pursuit of an optimal solution for drone detection. This thesis utilized CEDIA's High-Performance Computing (HPC) capabilities, made accessible through its collaboration with Red CUDI, to train the YOLOv8 algorithm. The evolution of the YOLO algorithm, spanning from YOLOv4 to YOLOv8, reflects ongoing enhancements and feature enrichments. Below is a detailed description of the training process for the different versions of YOLO.

#### Training with YOLOv4

YOLOv4 was the most challenging version to implement for various reasons. Detection of test images was performed locally, and it was also the first network being tested. The network training is conducted online using Google Colaboratory. Consequently, each time it connects to the Colab workspace, it requires rerunning all the cells (See. Annex A.1).

#### Training with YOLOv5s

As previously mentioned, in the case of YOLOv5, the 's' model was used, which is more compact and offers several advantages compared to larger versions. This model was trained and tested entirely in the cloud, following the steps provided by Google Colab itself. Therefore, this training process was much simpler than the previous version, as upon entering Colab, a guide and information on how to use it are readily available (See. Annex B.1).

#### Training with YOLOv7s

In the case of YOLOv7, the 'tiny' model was utilized, which is the smallest and offers several advantages compared to the larger versions. This model was trained and tested entirely in the cloud, following the steps provided by Google Colab itself. Therefore, this training process was much simpler than previous versions, as upon entering Colab, a guide and information on how to use it are readily available (See. Annex C.1).

#### Training with YOLOv8s

In the case of YOLOv8, "tiny" model was utilized, which is the smallest and offers several advantages compared to the larger versions. This model was trained and tested entirely in the cloud, following the steps provided by Google Colab itself. Therefore, this training process was much simpler than previous versions, as a guide and information on how to use it are readily available upon entering Colab (See. Annex D.1).

## Training YOLOv8 at CEDIA

The training of YOLOv8 at CEDIA has enhanced the model's ability to detect and classify objects in urban environments. Specifically, the model excels in detecting small objects like people and animals, as well as objects that are challenging to distinguish, such as cars and motorcycles. YOLOv8 trained at CEDIA has found applications in various fields, including public safety, surveillance, and task automation (See. Annex E.1, F.1).

### 4.2.4 Metrics

In evaluating the YOLOv8 component of our Anti-Drone Prototype System, we employ a range of key metrics to assess its detection capabilities comprehensively. Some of the primary metrics include:

- IoU (Intersection over union). IoU is the most commonly used metric for evaluating the performance of object detection systems. It measures the ratio of intersection area between the detected bounding box and the ground truth bounding box [139].
- AP (Average precision). AP measures the area under the precision-recall curve. A precision-recall curve is a graphical representation of the precision and recall of the system for different IoU thresholds [139].

- mAP (mean average precision). mAP is the mean of the AP values for all classes in the dataset. A high mAP value indicates that the system has good performance for all classes in the dataset [139].
- F1-score. The F1 score is a combination of precision and recall. It is calculated as the harmonic mean of precision and recall [139].
- Precision-recall curve. The precision-recall curve is a graphical representation of the precision and recall of the system for different IoU thresholds [139].

For the evaluation of drone detection systems using YOLO variants, we choose mAP50(B) and mAP50-95(B) as our metrics [140, 141, 142]. mAP50(B) is chosen as a standard overlap threshold that offers a balance between accuracy and tolerance in detection. This threshold indicates that detections are considered correct if the overlap area (IoU) between the predicted bounding boxes and the true ones is at least 50%. This level of overlap is significant in practical applications where moderate precision in localization is sufficient to consider that an object, such as a drone in this case, has been correctly detected. The choice of this threshold is motivated by the need to assess the prototype system's ability to reliably identify the presence of drones, even under conditions where the precision of localization might be compromised by factors such as the drone's rapid movement or limitations in image resolution. On the other hand, mAP50-95(B) is adopted to provide a more rigorous and comprehensive evaluation of detection performance, by including a wider range of overlap thresholds from moderate (0.5) to highly precise (0.95). This composite metric underscores the importance of precision in the localization of predicted bounding boxes, essential for situations requiring precise identification and neutralization of drones. The inclusion of this broad range allows for a more detailed understanding of how the YOLO model handles variations in detection precision, which is critical for assessing its applicability in security scenarios where drones may pose threats at different ranges and angles of approach. These selected metrics will allow us to quantitatively evaluate the system's ability to accurately identify and localize drones across a range of overlap thresholds, providing a holistic view of its overall detection performance in various scenarios and against different sizes of drones.

## 4.3 Counter Drone Threats: Implementing Jamming Technology

One such countermeasure gaining attention is the Realtek RTL8814AU (see Figure 4.2), a powerful jamming solution designed to address the escalating threat of drones. This section delves into the features, working principles, applications, and implications of Realtek RTL8814AU as a formidable jamming technology in the context of countering drone

| Parameter | Value |
|---|---|
| Chipset | Realtek RTL8814AU |
| WiFi Standards | IEEE 802.11ac/a/b/g/n |
| WiFi Frequency | Dual Band 2.4GHz or 5GHz |
| Antenna Connector | RP-SMA female x 4 |
| Antenna Type | 2.4G/5GHz Dual-Band 5dBi dipole antenna |
| Wireless Performance | 802.11a: up to 54Mbps<br>802.11b: up to 11Mbps<br>802.11g: up to 54Mbps<br>802.11n: up to 300Mbps<br>802.11ac: up to 867Mbps |
| Wireless Security | 64/128 bit WEP,WPA/WPA2,WPA-PSK/WPA2-PSK,WPS |
| Interface | USB 3.0 |
| OS Requirement | Windows XP, Vista, 7, 8/8.1 and Windows 10 32/64bit,<br>macOS 10.5 to 10.14 or later<br>Linux |

Table 4.2: Alfa AC1900 WiFi Adapter: 1900 Mbps, long-range, dual-band, USB 3.0, 4x 5dBi antennas.

threats. The Realtek RTL8814AU is a high-gain, long-range wireless network adapter with the capability to transmit and receive wireless signals over extended distances. Its key features include a detachable external antenna, compatibility with various operating systems, and support for multiple wireless standards such as IEEE 802.11ac. Additionally, it operates in the 2.4GHz and 5GHz frequency bands, allowing for versatility in jamming different types of drones. Table 4.2 summarizes all the characteristics of the antenna Realtek RTL8814AU.

Jamming technology, like the Realtek RTL8814AU, operates by emitting radio frequency (RF) signals on the same frequency band used by drones for communication with their remote controllers. This interference disrupts the connection between the drone and its operator, rendering the drone unable to receive commands or transmit data. The Realtek RTL8814AU is particularly effective due to its high transmission power, enabling it to overpower drone signals even at considerable distances. The Realtek RTL8814AU finds applications in various contexts, primarily for countering drone threats. It is deployed in areas where the unauthorized use of drones poses risks, such as airports, government facilities, critical infrastructure, and public events. By neutralizing drones in these environments, the technology ensures safety, security, and compliance with regulations. The use of jamming technology, including the Realtek RTL8814AU, raises legal and ethical considerations. Jamming RF signals can potentially interfere with legitimate wireless communications beyond drones, impacting nearby devices and networks. This necessitates responsible and regulated use of such technology, with adherence to relevant laws and guidelines.

Figure 4.2: Alfa AC1900 WiFi Adapter - 1900 Mbps 802.11ac Long-Range Dual Band

### 4.3.1 Process of a wireless Deauthentication Attack

As you can see in Figure 4.3, the remote controller of a drone initiates communication with the drone through a sequence of control and data messages exchanged between both devices [143]. The process unfolds as follows:

- Authentication request: The remote controller sends an authentication request message to the drone. This message informs the drone that the controller wishes to establish a connection and requests authentication from the device.
- Authentication response: The drone receives the authentication request and responds with an authentication response message. In this message, the drone may confirm the authentication request and provide the necessary details for the authentication process.
- Association request: Once authentication is completed, the remote controller sends an association request to the drone. This request informs the drone that the controller wishes to establish an association with the device to allow bidirectional communication.

- Association response: The drone receives the association request and responds with an association response message. In this message, the drone may confirm the association with the remote controller and provide the necessary details for subsequent communication.

- Data transmission: Once the association is established, both the remote controller and the drone can exchange control and telemetry data. This data may include flight commands, position information, battery status, and sensor data.

However, a jammer may disrupt the established connection between the drone and the remote controller through a deauthentication attack [144, 145]. This process involves the transmission of falsified deauthentication messages to interrupt communication between both devices. The attack process could follow these steps:

- The jammer, possibly within the signal range of the drone and the remote controller, employs specialized tools to send forged deauthentication packets [146]. These packets are directed to both the drone and the remote controller and are designed to make them believe that the other device has requested disconnection. As a result, the drone and the remote controller may lose the established connection and become unable to communicate effectively.

- Effects on communication: Once the deauthentication attack is initiated, the drone may lose the ability to receive commands from the remote controller, potentially resulting in loss of flight control. Similarly, the remote controller may cease to receive telemetry data from the drone, making it difficult for the pilot to monitor the drone's status and position. This could lead to a complete loss of drone control, potentially resulting in accidents or material losses.

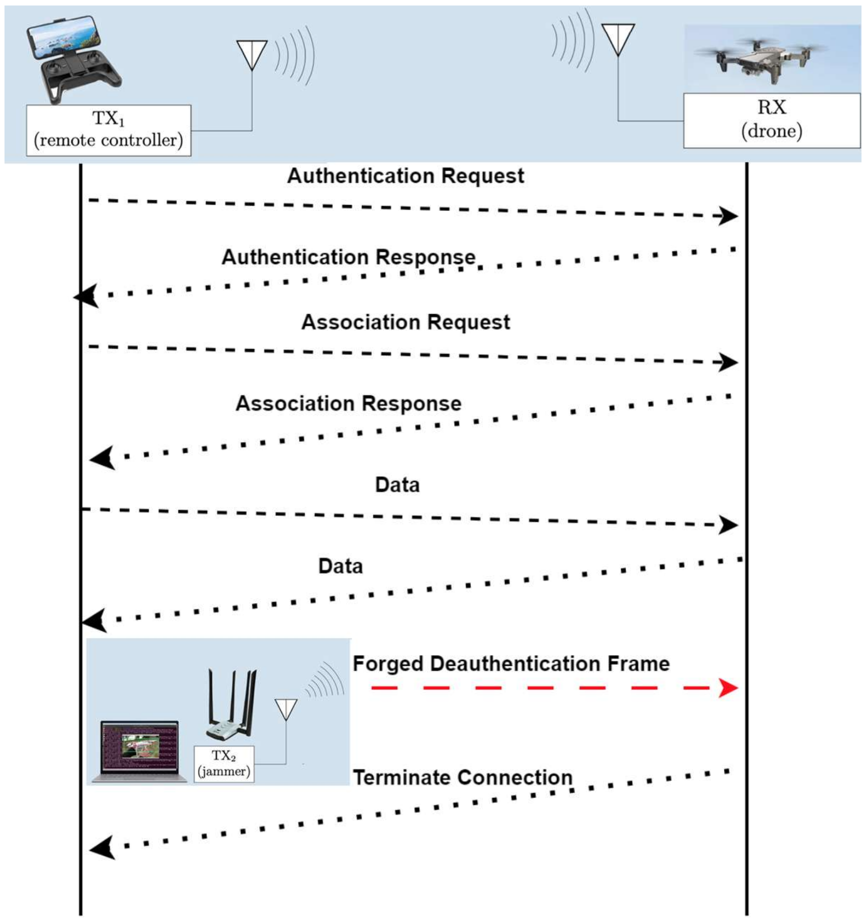


Figure 4.3: Process of a wireless deauthentication attack

### 4.3.2 Process of a wireless Disassociation Attack

A Disassociation attack in the wireless communication between a drone and its remote control entails the deliberate transmission of disassociation frames to the drone [92]. These frames are meticulously crafted to deceive the drone into perceiving a voluntary disconnection from the remote control. Consequently, the drone autonomously severs its association with the remote control, resulting in a disconnected state until manual intervention is initiated to re-establish the link. Unlike Deauthentication attacks, Disassociation attacks do not trigger an automatic reconnection process [147]. This aspect sets Disassociation attacks apart as potentially more disruptive, necessitating manual reconnection procedures to regain control over the drone. For a detailed illustration of the Disassociation attack mechanism, see Figure 4.4.

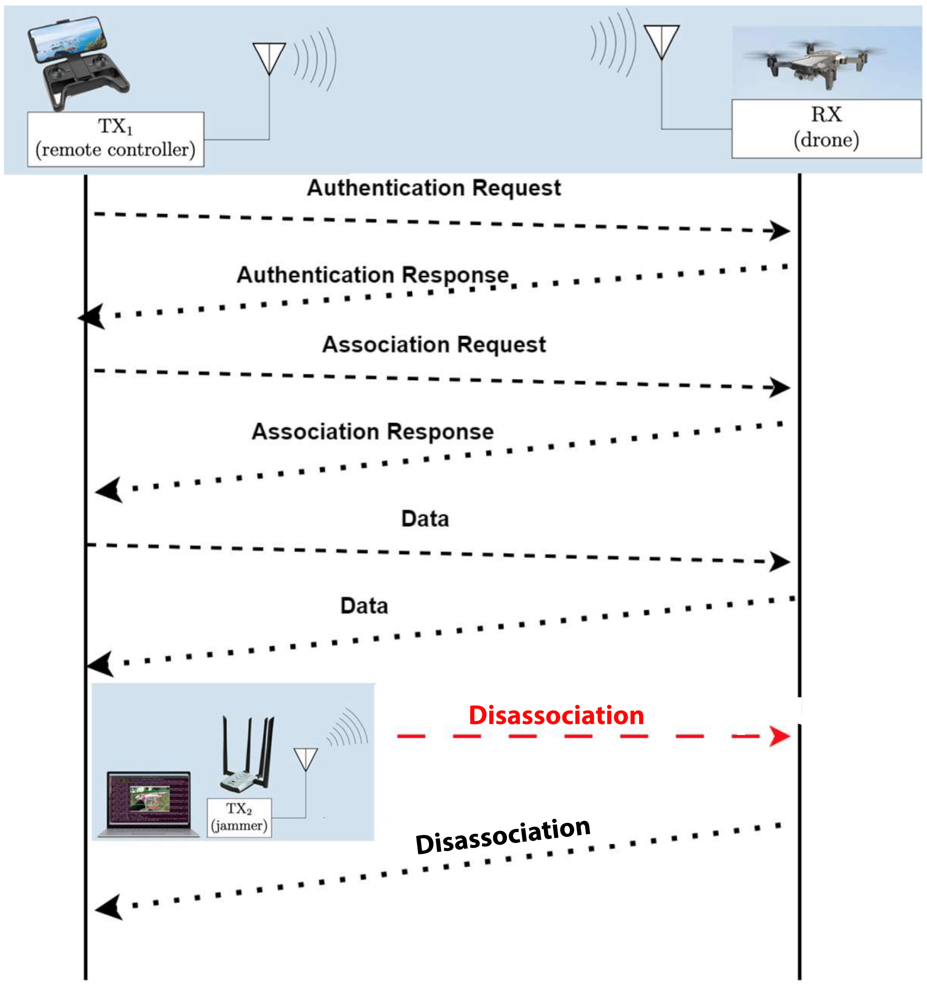


Figure 4.4: Process of a wireless disassociation attack

### 4.3.3 Packet injection

The Realtek RTL8814AU with packet injection capabilities can be used to launch a deauthentication attack on a WiFi network in drones. Packet injection is a technique that allows the jammer to send data packets to a network. In the case of a deauthentication attack, the jammer would send a deauthentication message to the drone. To carry out a deauthentication attack with a wireless card on a drone, the attacker must follow these steps:

1. Identify the target WiFi network. The jammer needs to identify the target WiFi network and the MAC address of the drone.

2. Obtain the MAC address of the target client. The jammer can obtain the MAC address of the target client through a packet scanning attack.

3. Generate a deauthentication message. The jammer must generate a valid deauthentication message directed at the target client.

4. Send the deauthentication message to the drone. The Jammer needs to send the deauthentication message to the access point using the wireless card.

Once the jammer has sent the deauthentication message, the drone will be disconnected from the network. The jammer can then attempt to connect to the network using the client's authentication key. This can allow the neutralization system to take control of the drone or interfere with its operations.

### 4.3.4 MDK tool for exploiting IEEE 802.11 protocol weaknesses.

MDK4 (Murder Death Kill 4) works by sending deauthentication packets to target devices within a WiFi network, causing them to disconnect from the network temporarily. Here is a overview of how MDK4 works:

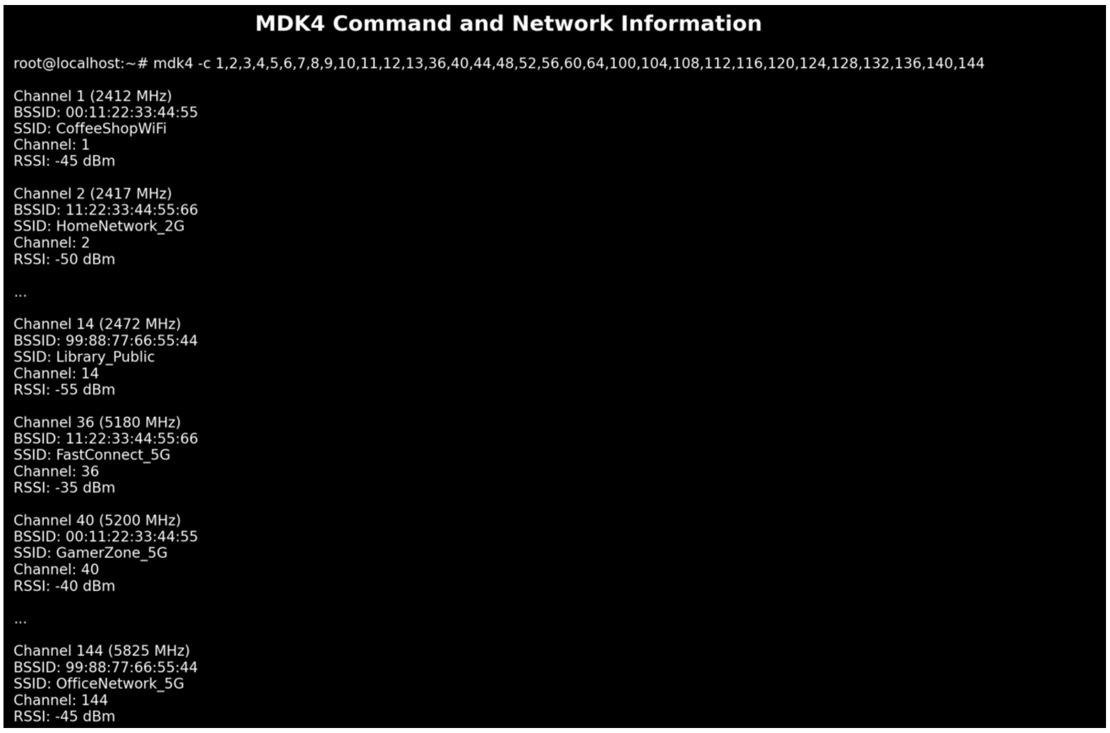
1. **Launch MDK4**: Open a terminal in Kali Linux and run the MDK4 tool with the appropriate parameters to specify the target WiFi network and interface. The command typically looks like this:

   ```
   mdk4 wlan0mon d -bssid [target_BSSID] -c [target_channel]
   ```

   - wlan0mon is the name of your wireless interface in monitor mode.
   - -bssid [target_BSSID] specifies the MAC address of the target access point.
   - -c [target_channel] specifies the channel the target network is operating on.

2. **Deauthentication and disassociation attack**: MDK4 will start sending deauthentication and disassociation packets to all devices connected to the target network. These packets appear as if they are coming from the access point itself, instructing the devices to disconnect and then attempt to reconnect.

3. **Monitor results**: As MDK4 sends deauthentication packets, we can monitor the results in the terminal. As you can see in Figure 4.5, we can see messages indicating which devices have been deauthenticated.



MDK4 Command and Network Information

```
root@localhost:~# mdk4 -c 1,2,3,4,5,6,7,8,9,10,11,12,13,36,40,44,48,52,56,60,64,100,104,108,112,116,120,124,128,132,136,140,144

Channel 1 (2412 MHz)
BSSID: 00:11:22:33:44:55
SSID: CoffeeShopWiFi
Channel: 1
RSSI: -45 dBm

Channel 2 (2417 MHz)
BSSID: 11:22:33:44:55:66
SSID: HomeNetwork_2G
Channel: 2
RSSI: -50 dBm

...

Channel 14 (2472 MHz)
BSSID: 99:88:77:66:55:44
SSID: Library_Public
Channel: 14
RSSI: -55 dBm

Channel 36 (5180 MHz)
BSSID: 11:22:33:44:55:66
SSID: FastConnect_5G
Channel: 36
RSSI: -35 dBm

Channel 40 (5200 MHz)
BSSID: 00:11:22:33:44:55
SSID: GamerZone_5G
Channel: 40
RSSI: -40 dBm

...

Channel 144 (5825 MHz)
BSSID: 99:88:77:66:55:44
SSID: OfficeNetwork_5G
Channel: 144
RSSI: -45 dBm
```

Figure 4.5: MDK4 Command and Network Information

## 4.4 Prototype

The need to safeguard critical spaces against unauthorized drone use has become a priority. In response to this issue, an anti-drone prototype system has been developed, combining the effectiveness of YOLOv8 for drone detection with the power of the ALFA AWUS036ACS wireless card and Kali Linux's MDK4 software for launching deauthentication attacks. This section will explore the architecture, functionality, and potential of this system in the fight against drone threats.

### 4.4.1 System Architecture

The prototype consists of three main elements: a camera for real-time image and video capture, a processing unit that runs YOLOv8 for detection, and an ALFA AWUS036ACS wireless card responsible for neutralization. The processing unit uses YOLOv8 to analyze the image stream for drones, while the wireless card handles deauthentication attacks when an unauthorized drone is detected (See Figure 4.6 ).

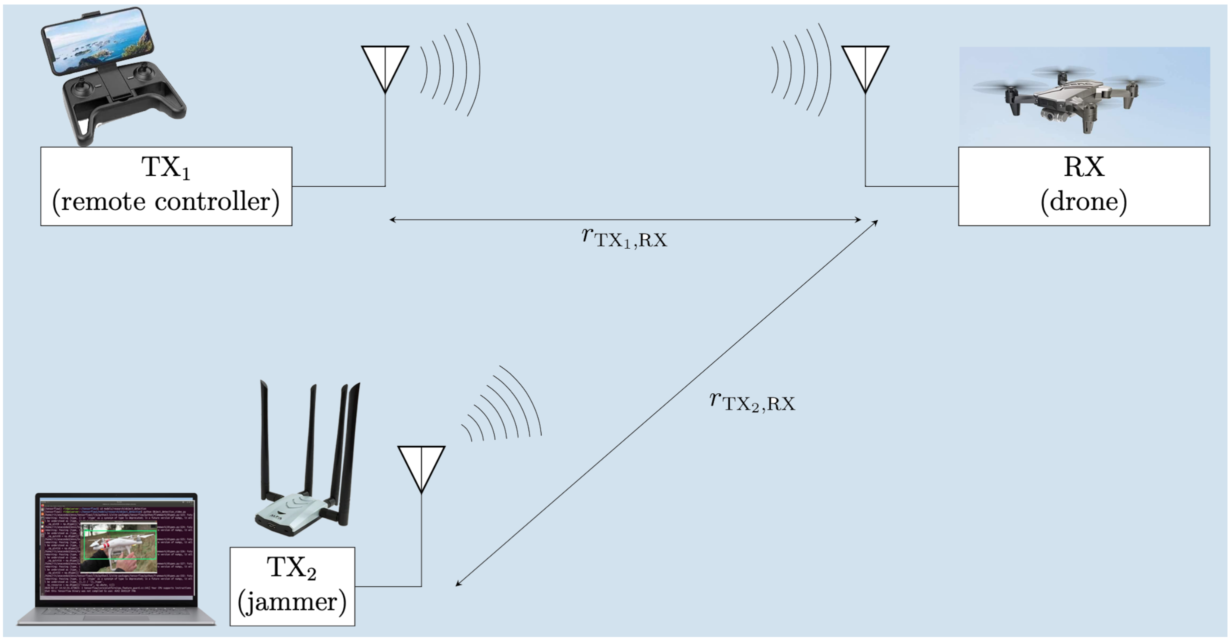


Figure 4.6: Prototype

### 4.4.2 Drone Detection with YOLOv8:

Drone detection utilizing YOLOv8 leverages the robust capabilities of this real-time object detection algorithm to identify and classify drones within visual data streams. The process typically involves the following steps:

- Input acquisition: Initially, visual input data, such as images or video frames captured by cameras or sensors, are acquired for analysis.
- Preprocessing: The input data undergo preprocessing to enhance its quality and suitability for detection tasks. This may involve resizing, normalization, or other image enhancement techniques.
- Object detection: YOLOv8 applies a deep CNN to the preprocessed input data. This network is trained on vast datasets to recognize objects of interest, including drones, within images or video frames.
- Prediction: The CNN within YOLOv8 generates predictions for each grid cell in the input data. For every detected object, the algorithm predicts bounding box coordinates and assigns a confidence score indicating the likelihood of the detected object being a drone.

- Thresholding: Predictions with confidence scores below a predefined threshold are discarded to filter out false positives and ensure reliable drone detection.

- Post-processing: The remaining predictions are refined through post-processing techniques such as non-maximum suppression to eliminate redundant detections and retain only the most confident and accurate drone detections.

- Visualization: Finally, the detected drones and their bounding boxes are visualized or reported, providing actionable information for decision-making or further analysis.

### 4.4.3 Jammer

To neutralize detected drones effectively, our prototype combines the ALFA AWUS036ACS wireless card's packet injection capability with MDK4, a specific tool in Kali Linux. This integration enables the system to launch deauthentication attacks on drones' WiFi networks, disrupting their connection with operators. By promptly initiating these attacks upon drone detection, the system ensures timely neutralization of the threat. The seamless integration of YOLOv8 for drone detection with the ALFA AWUS036ACS wireless card and MDK4 for neutralization presents a promising approach in combating drone threats. This prototype achieves a balance between precise detection and swift neutralization, bolstering public safety and safeguarding critical infrastructure against unauthorized drone use. By disrupting the connection between the drone and its operator using forged deauthentication frames, the system effectively neutralizes the drone threat. This immediate response hampers the drone's ability to communicate and receive commands, thereby mitigating potential risks. By seamlessly integrating detection and neutralization capabilities, the system swiftly and effectively counters unauthorized drone activities, enhancing overall security measures. Appendix I provides photographs of the functional anti-drone system prototype.

# Chapter 5

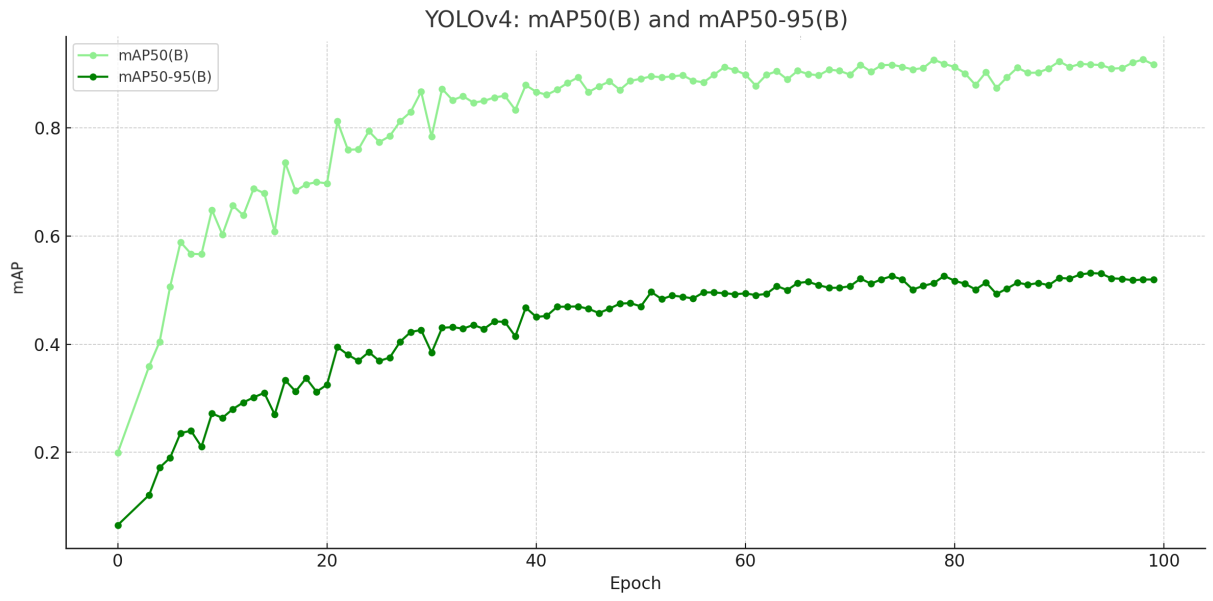
# Results and Discussion

The development of the Anti-Drone Prototype System, which combines YOLOv8 for drone detection with the ALFA AWUS036ACS wireless card and MDK4 for neutralization, marks a significant advancement in countering the threats posed by drones. This chapter presents the results obtained through rigorous testing and explores the implications and potential applications of this innovative system. The focus here is on the performance of the prototype, its accuracy in detection, and its effectiveness in neutralization, which are pivotal in ensuring public safety and safeguarding critical infrastructure.

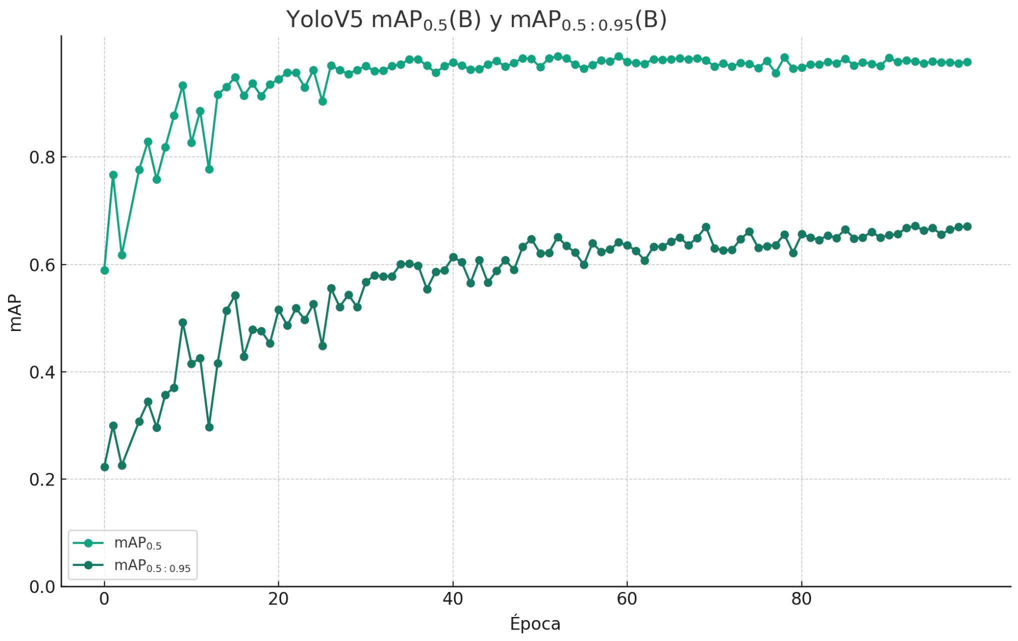
## 5.1 Evaluation of drone detection systems based on YOLO

In exploring the effectiveness and efficiency of convolutional neural networks, specifically within the YOLO variants framework, particular emphasis is placed on the quantitative evaluation of object detection metrics, such as the Mean mAP. This metric provides a comprehensive assessment of the correspondence between predicted and annotated bounding boxes, with a specific focus on two critical overlap thresholds: mAP50(B) and mAP50-95(B).

In Figure 5.1a, the trajectory of these metrics across 100 training epochs is observed, providing vital insights into the model's ability to predict bounding boxes accurately and reliably. In a training environment like Google Colab, training times are influenced by various factors, including image resolution and hyperparameter settings. Pragmatically, it is reasonable to anticipate that the model may begin to show signs of convergence around epoch 50, taking approximately 30 minutes to reach this point, depending on the specific environment and model configuration. Additionally, it should be noted that this version of YOLO significantly reduces the size of the images to 416x416 pixels. This reduction is

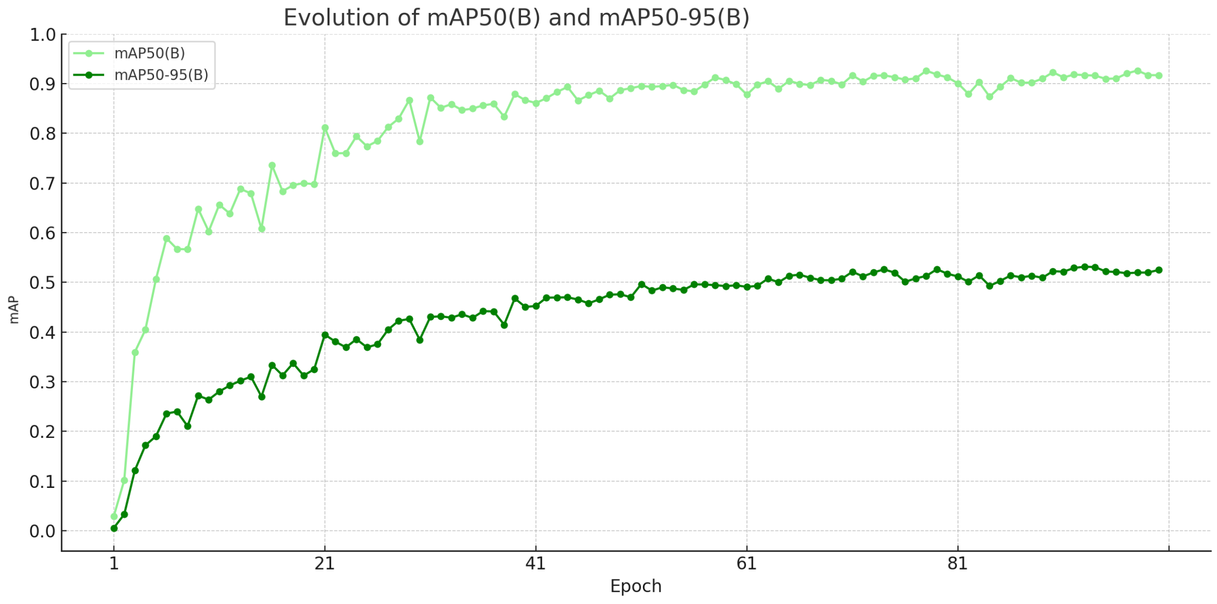


(a) YOLOv4

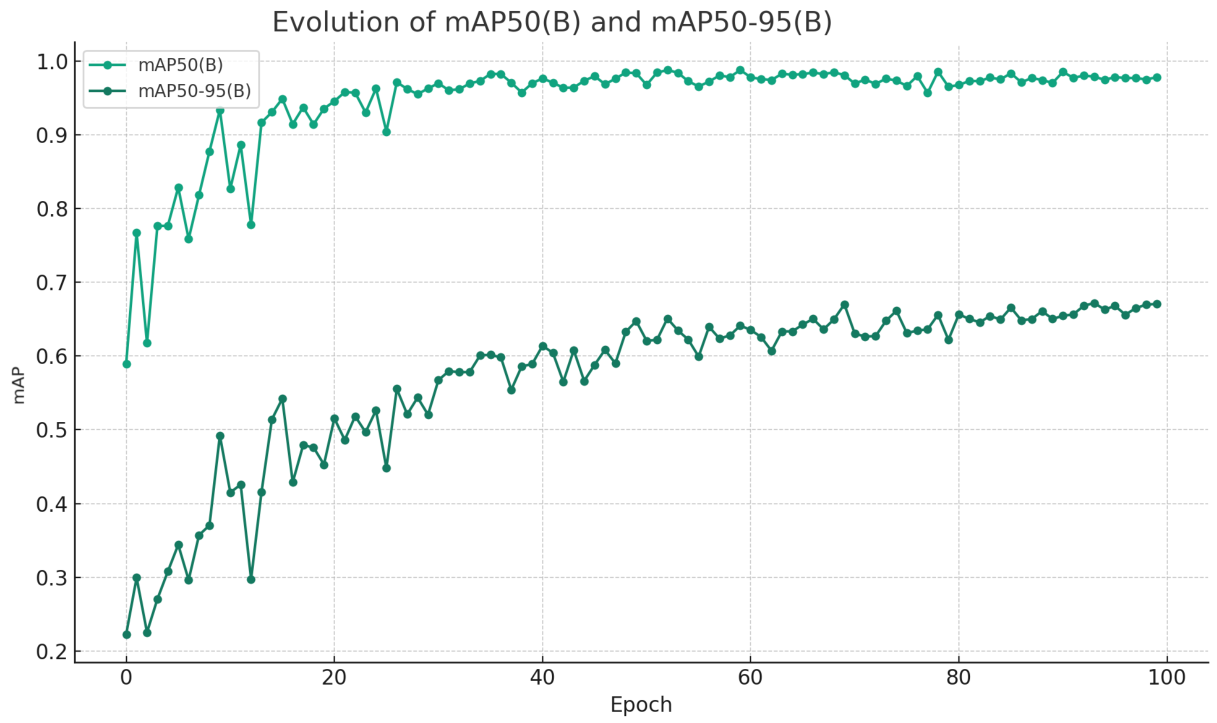


(b) YOLOv5s

Figure 5.1: Evolution of $\text{mAP}_{0.5}(B)$ and $\text{mAP}_{0.5:0.95}(B)$ through 100 epochs for YOLOv4 and YOLOv5



(a) YOLOv7



(b) YOLOv8

Figure 5.2: Evolution of $\mathrm{mAP}_{0.5}(B)$ and $\mathrm{mAP}_{0.5:0.95}(B)$ through 100 epochs for YOLOv7 and YOLOv8.

made to balance the need for detailed detection with computational efficiency. However, this process also entails the loss of information that could be useful for detection.

The implementation of YOLOv5s has proven to be remarkably efficient, particularly regarding the network training times, providing a robust platform for model evaluation using a specific dataset. The mAP metric, with a particular focus on the $\mathrm{mAP}_{0.5}(B)$ and $\mathrm{mAP}_{0.5:0.95}(B)$ thresholds, stands at the center of this analysis, providing a comprehensive insight into the model's accuracy and reliability in predicting bounding boxes. Figure 5.1b shows the trajectory of the mAP metrics across 100 training epochs, highlighting the model's capability to predict bounding boxes accurately and reliably. This analysis was conducted using a batch size of 32 and a single, specifically created dataset, The training duration is a crucial component, especially considering the model implementation on platforms like Google Colab. With an estimated training time of approximately 23 minutes for 100 epochs, YOLOv5s demonstrates notable computational efficiency, allowing for more iterative testing and adjustments within a reduced timeframe. The model was trained using 416x416 pixel images, thereby balancing detailed detection and computational efficiency. The chosen resolution, although optimal for this scenario, might be subject to future evaluations and adjustments, depending on the specific needs of the target application.

The deployment of YOLOv7 has been observed to bring about commendable proficiency, particularly in the context of network training times, offering a sturdy platform for model evaluation using a distinct dataset. The mAP metric, with specific emphasis on the $\mathrm{mAP}_{0.5}(B)$ and $\mathrm{mAP}_{0.5:0.95}(B)$ thresholds, is pivotal in this analysis, providing a holistic view of the model's capability to predict bounding boxes with accuracy and reliability. The evolution of the mAP50(B) and mAP50-95(B) metrics during the training epochs in YOLOv7 exhibits an upward trend, indicating a progressive improvement in the model's detection capability. In particular, mAP50(B) reflects effective feature capture for detections with moderate overlap, while mAP50-95(B) reveals the model's competence in sustaining detection accuracies from moderate to extremely precise levels. The trajectory of these metrics, especially the speed at which the model converges and its stability after reaching a peak, offers critical insights into the model's robustness and its ability to generalize learnings across various IoU thresholds. Any variation, stability, or fluctuation in these metrics over epochs not only indicates the model's efficiency but also points out areas that could benefit from future adjustments in hyperparameters or training strategies. The training of the YOLOv7 model, conducted in a Google Colab environment, was carried out using a batch size of 32 and an image resolution of 416x416 pixels, which was chosen to balance detection accuracy with computational efficiency. The total training time for 100 epochs was approximately 4 hours and 50 minutes, highlighting the time efficiency of YOLOv7, especially when compared to previous versions of the YOLO architecture.

The YOLOv8 object detection model demonstrates the capability to detect drones with a precision of up to 90%, as measured by the mean average precision (mAP) at the image-level class (AP) (See Figure 5.3). Training a YOLOv8 model for drone detection is a timeconsuming process, and it can take several hours to even days to complete. During the initial training phase (epochs 0-33), the YOLOv8 model exhibits sharp learning ability, evidenced by a significant rise in the mAP50(B) and mAP50-95(B) metrics, indicating effective pattern recognition within the dataset. Throughout the middle phase (epochs 34-66), the metrics show fluctuations along with gradual improvement, suggesting model adaptation to dataset complexities and possibly requiring a review of the learning rate and regularization strategies to mitigate oscillations in learning. The final phase (epochs 67-100) displays persistence in metric fluctuations and a less pronounced improvement trend, implying the need to explore techniques for stabilizing training, verifying model robustness, and ensuring that the model is not oscillating around a local minimum in the solution space. Throughout these phases, continuous evaluation of the model's performance on a validation set is imperative to ensure the generalization of observed metrics and guide optimization strategies in future training iterations. This iterative process is essential for refining the YOLOv8 model's drone detection capabilities and achieving high precision in real-world applications.

## Analysis of YOLO Models

Table 5.1 presents a comparison of various YOLO versions. Examining the results, YOLOv8s stands out in terms of precision and recall, with an mAP (IoU 0.5) of 96% and a recall of 98%, making it the model with the best detection capability. However, this efficiency comes at the cost of significantly longer per-epoch training time and a slightly larger model size compared to, for example, YOLOv5s. On the other hand, YOLOv5s provides what appears to be an optimal balance between training efficiency and model size, being the smallest model and relatively quick to train. Although its mAP (IoU 0.5:0.95) is slightly lower than that of YOLOv8s, its smaller size and shorter training time can make it more applicable for situations with resource and time constraints. Therefore, the selection of the appropriate model may heavily depend on the specific use case, where YOLOv8s might be preferred for applications where precision is critical and resources are not a concern, while YOLOv5s could be more suitable for applications in resource-constrained environments.

| Model | Epochs | mAP (IoU 0.5) | mAP (IoU 0.5:0.95) | Recall | Training Time per Epoch (s) | Model Size (MB) |
|---|---|---|---|---|---|---|
| YOLOv4s | 100 | 89% | 55% | 95% | 30 | 396 |
| YOLOv5s | 100 | 91% | 54% | 96% | 43 | 55 |
| YOLOv7s | 100 | 92% | 53% | 97% | 4 hours 50 minutes | 60 |
| YOLOv8s | 100 | 94% | 68% | 98% | 6 hours 30 minutes | 62 |

Table 5.1: Performance and size metrics for different YOLO model versions.

## Model Selection and Training with YOLOv8

After evaluating different iterations of the YOLO architecture, YOLOv8 was selected for its enhanced mean Average Precision (mAP) and recall metrics, as outlined in Table 5.1 Despite its increased training duration per epoch and marginally larger model size relative to previous versions, the accuracy of object detection is paramount for our application, making the trade-off acceptable. The training was executed on the CEDIA supercomputer, which offered substantial computational power, enabling a deeper and more thorough model tuning and validation process. Utilizing such a powerful computing environment facilitated the refinement of the model, achieving notable precision and recall in our assessments. The empirical results from the supercomputer training sessions corroborated our selection of YOLOv8, establishing it as a formidable and precise tool for object detection tailored to our specific requirements.

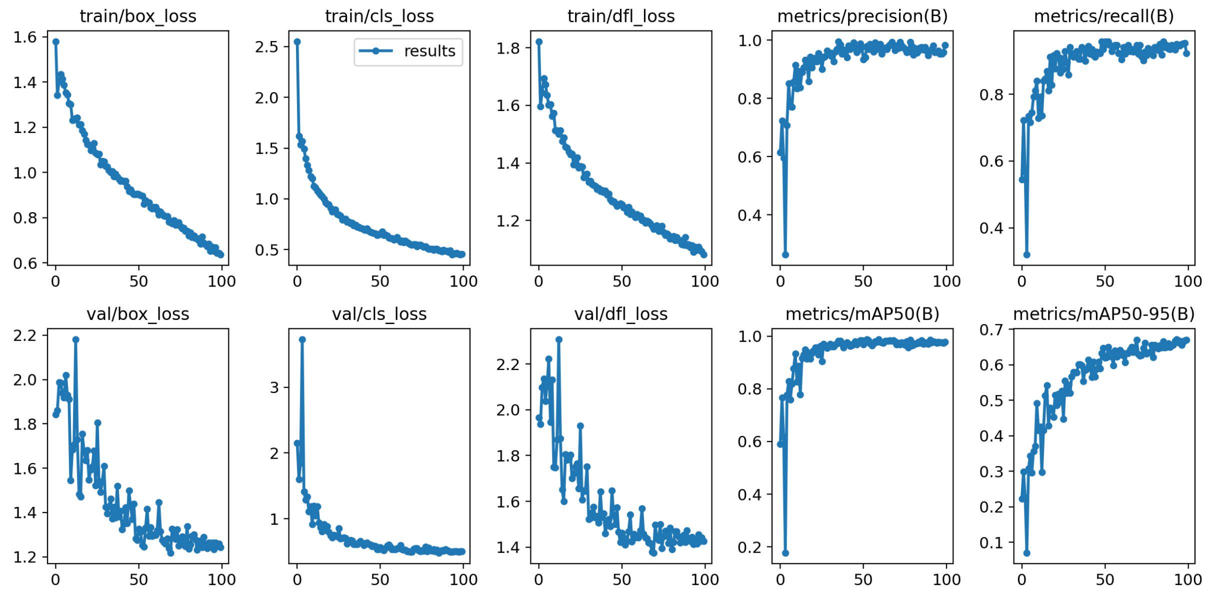


Figure 5.3: YOLOv8 training.

Figure 5.3 shows the performance convergence of a YOLO-based drone detection system, evidenced by the progression of key metrics over 100 training and validation epochs. A consistent reduction in losses for bounding box accuracy, class identification, and edge definition signifies the model's growing adeptness in pinpointing and distinguishing drones. The model sustains a high level of precision, and an early upward trend in recall levels off, indicating its strengthening capability to accurately spot drones while avoiding misses. Average precision metrics across a spectrum of IoU thresholds show the model's maturing ability to detect drones with varying degrees of precision regarding the overlap of predicted and actual bounding boxes. Collectively, these graphs manifest robust model training and validation, with advancement toward high reliability in drone detection—an essential aspect for security and monitoring applications employing YOLO.

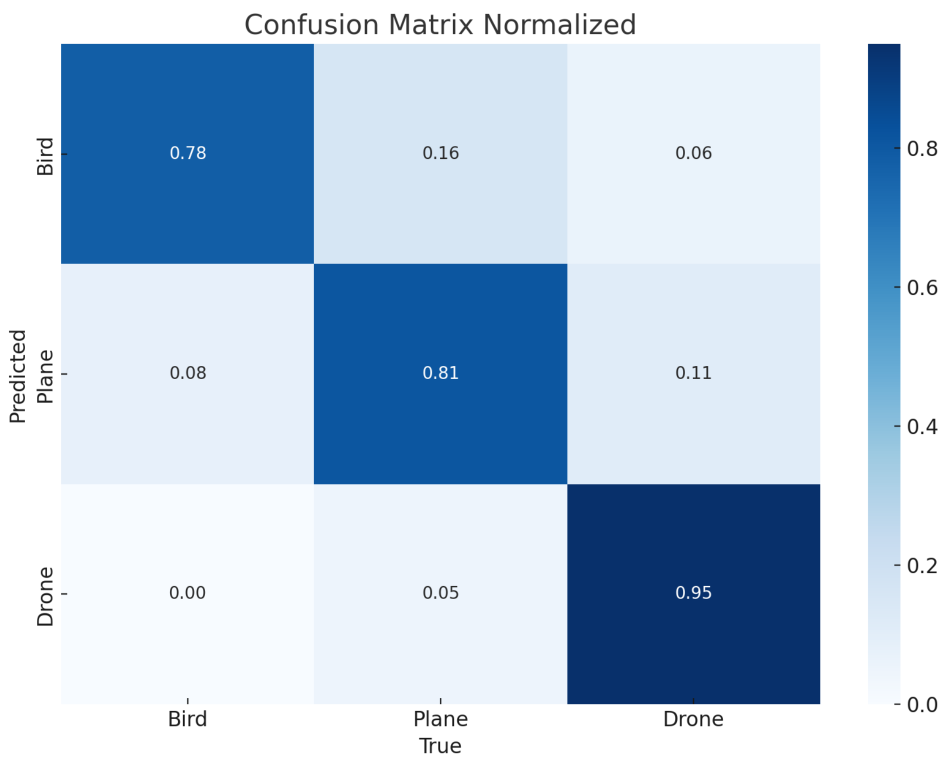


Figure 5.4: Normalized confusion matrix of 100 epochs of training

Figure 5.4 depicts the normalized confusion matrix for the YOLO-based classification model focused on the identification of drones as opposed to birds and planes. This visual representation underscores a true positive rate of 95% for drone detection, signifying the model's high reliability in this particular category. The absence of false negatives for misclassifying birds as drones further emphasizes the model's effectiveness in differentiating drones from other non-manned flying objects. While the false negative rate of 11% for drones incorrectly classified as planes and the false positive rate of 5% for planes mistaken as drones suggest areas for improvement. This figure highlights the model's success in drone detection within the realm of applied computer vision, which is crucial for aerial security and airspace management. The detailed metrics confirm the robustness of the YOLO model in the drone category and provide a foundation for ongoing optimization in the classification of planes and birds.

## 5.2 Integration of Drone Detection and Neutralization Systems for Immediate Response

The integration of drone detection and neutralization systems represents a proactive and effective approach to safeguarding airspace and critical assets from the evolving threats posed by drones. In this context, two areas have gained significant attention due to their impact and relevance across multiple applications: advanced object detection using computer vision algorithms, such as drone detection, and the security of Wi-Fi wireless networks, particularly in the context of stress testing and deauthentication. This section focuses on a detailed technical exploration of these two aspects, covering everything from practical implementation to the underlying theoretical foundations (See. Annex G.1).

### 5.2.1 Work Environment Preparation

Our proposal is written in Python and uses the OpenCV computer vision library, PIL (Python Imaging Library) for image processing, and YOLO, a real-time object detection algorithm [148].

- Tkinter. It is used for the graphical user interface (GUI). Tkinter is a standard Python library for creating graphical interfaces.
- OpenCV (cv2). It is Used for image and video processing and analysis. OpenCV is a widely used computer vision library.
- PIL (Python Imaging Library). It is used for image operations, such as opening and displaying images in the GUI.
- Pandas. It is used for data manipulation and analysis. In this case, it is used to handle the data from the detections made by YOLO.

To effectively neutralize detected drones, our system integrates a comprehensive workflow encompassing both detection and neutralization measures.

### 5.2.2 Workflow to detect drones

The code follows a specific workflow to detect drones in images or videos:

1. **Initialization**. An instance of the VideoApp class is created, which initializes the GUI, loads the YOLO model, and prepares the necessary variables.
2. **User interface**. The GUI allows the user to interact with the program, select video files, and control the display.

3. **Video capture**. Using OpenCV, the program captures video images in real-time (webcam) or from a selected video file.

4. **Image processing**. The captured images are processed using the YOLO model to detect objects. YOLO divides the image into a grid and predicts bounding boxes and class probabilities for each grid cell.

5. **Drone detection**. The YOLO model identifies objects in the image. If an object is classified as a drone (according to the classes defined in classes-file), specific logic is activated (such as enabling a button in the GUI).

The GUI allows the user to interact with the program intuitively. Drone detection is visualized in real-time, and the interface provides controls for managing video capture and analysis.

### 5.2.3 Drone Neutralization Workflow

The code follows a specific workflow to neutralize drones using MDK4:

1. **Activation of neutralization measures**: If neutralization is deemed necessary, the system activates countermeasures, such as deauthentication/disassociation attacks using MDK4.

   - **Packet spoofing**. MDK4 generates and transmits spoofed deauthentication/disassociation packets. These packets have a specific format, with fields indicating the sender and recipient of the deauthentication/disassociation. When received by a drone, they cause disconnection from the network. To send deauthentication/disassociation packets, the Wi-Fi card of the jammer must be in monitor mode, which allows listening and transmitting on any Wi-Fi channel.
   - **Automation with bash**: Our script automates the process of network identification, switching to monitor mode, and executing mdk4 to perform deauthentication attacks. It uses command line tools like `iwconfig` to manipulate network interfaces and `nmcli` to obtain information about available Wi-Fi networks.
   - **Parallel processes and channel management**. Our script starts mdk4 in parallel processes for each detected network, each operating on the channel corresponding to that network. This implies advanced process control and management of multiple channels simultaneously.

2. **Confirmation of neutralization**: Post-deployment, the system verifies successful drone neutralization by monitoring its behavior.

## 5.3 UAV Surveillance and Countermeasure Application

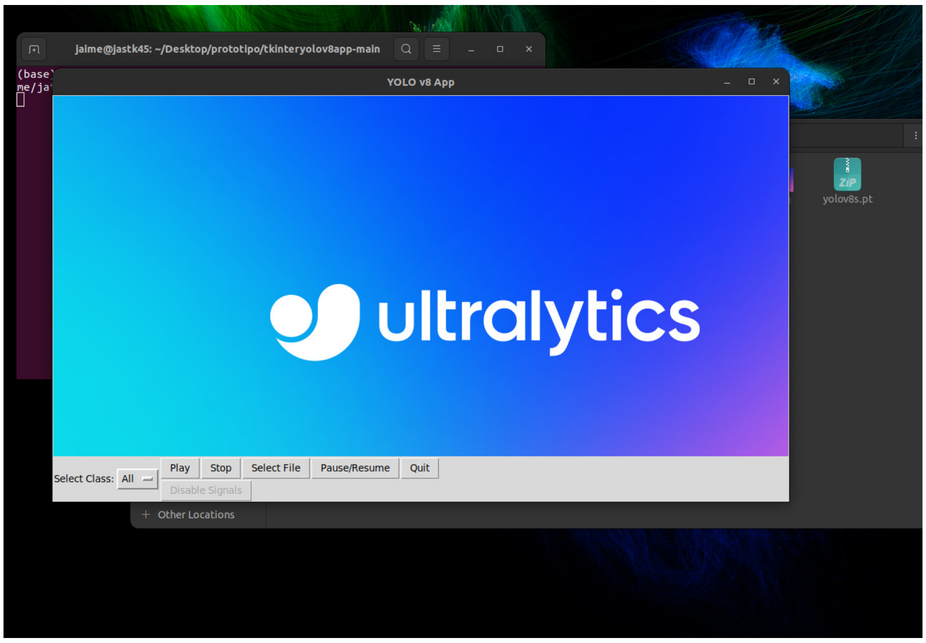
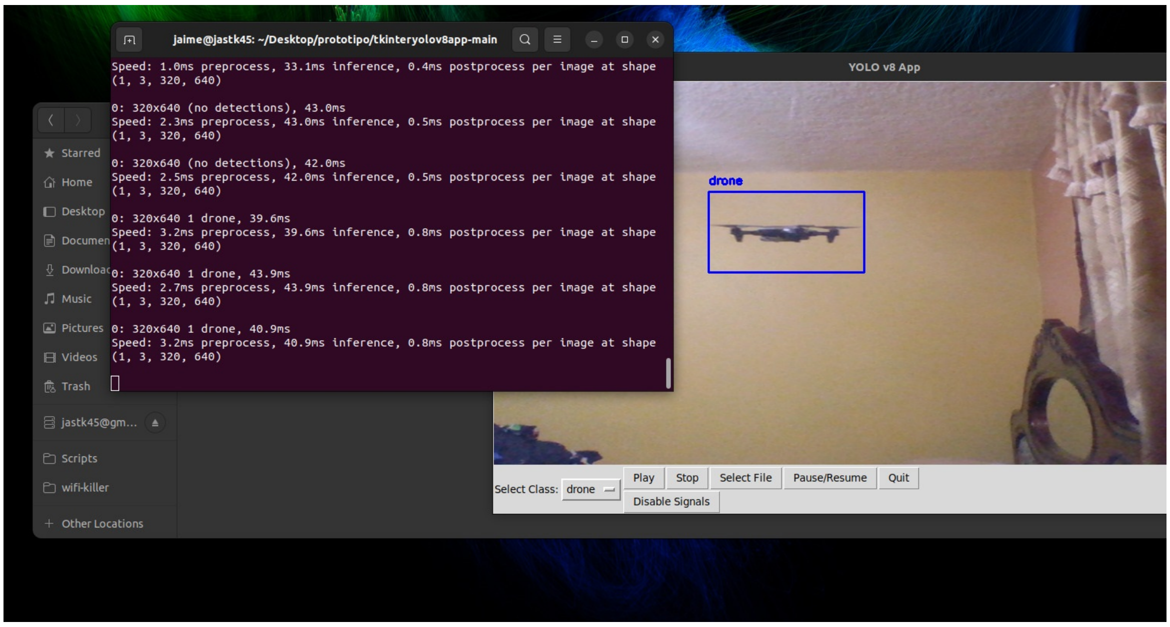


Figure 5.5: Comprehensive UAV Surveillance and Countermeasure Application

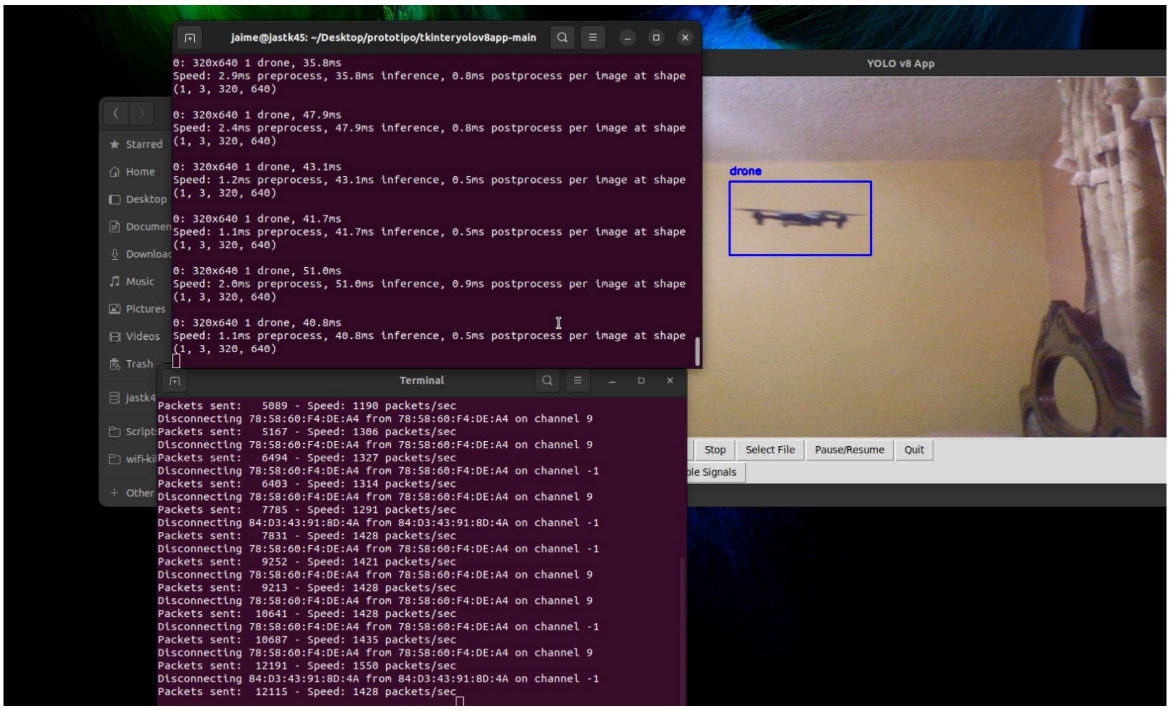
Figure 5.5 displays the graphical user interface (GUI) of the "YOLO v8 App", which is designed for object detection using the YOLO algorithm. At the bottom, there is a toolbar with several control buttons: `Play` to initiate processing or analysis, `Stop` to halt any ongoing activity, `Select File` for the user to choose a video or image file for processing, `Pause/Resume` to pause and subsequently continue the active process, and `Quit` to close the application. Adjacent to the control buttons is a dropdown menu labeled `Select Class:` with the current option set to `All`, allowing the user to select specific object classes for detection. There is also a button labeled `Disable Signals` that is in a disabled state, indicating that it is not available for use at that moment.



```
Speed: 1.0ms preprocess, 33.1ms inference, 0.4ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 (no detections), 43.0ms
Speed: 2.3ms preprocess, 43.0ms inference, 0.5ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 (no detections), 42.0ms
Speed: 2.5ms preprocess, 42.0ms inference, 0.5ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 39.6ms
Speed: 3.2ms preprocess, 39.6ms inference, 0.8ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 43.9ms
Speed: 2.7ms preprocess, 43.9ms inference, 0.8ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 40.9ms
Speed: 3.2ms preprocess, 40.9ms inference, 0.8ms postprocess per image at shape
(1, 3, 320, 640)
```

Figure 5.6: Integration of Drone Detection

Figure 5.6 shows our application running the "YOLO v8 Application" for object detection. The main focus of the application window is a live video or still image where a drone has been detected, as indicated by a blue bounding box around the object and a `drone` label above it. This demonstrates the real-time object detection capabilities of the application. On the left side of the screen, a terminal window displays a log of the detection process, including the speed of preprocessing, inference, and postprocessing for each processed frame or image. It shows that the application is running detections at a frame size of 320x640 pixels and lists the detection results, which in several entries correctly identify "1 drone". Below the video, there is a toolbar with drop-down menus and buttons that provide user interaction with the application. The dropdown is set to `drone`, indicating that the detection is filtered to this specific class. The buttons provided are labeled `Play`, `Stop`, `Select File`, `Pause/Resume`, and `Exit`, which are used to control the video streaming and detection process. There is also a button called `Disable signals`, which is only activated when detecting drones with a functionality to neutralize the detected drone.



```
0: 320x640 1 drone, 35.8ms
Speed: 2.9ms preprocess, 35.8ms inference, 0.8ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 47.9ms
Speed: 2.4ms preprocess, 47.9ms inference, 0.8ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 43.1ms
Speed: 1.2ms preprocess, 43.1ms inference, 0.5ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 41.7ms
Speed: 1.1ms preprocess, 41.7ms inference, 0.5ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 51.0ms
Speed: 2.0ms preprocess, 51.0ms inference, 0.9ms postprocess per image at shape
(1, 3, 320, 640)

0: 320x640 1 drone, 40.8ms
Speed: 1.1ms preprocess, 40.8ms inference, 0.5ms postprocess per image at shape
(1, 3, 320, 640)
```

```
Packets sent:   5089 - Speed: 1190 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:   5167 - Speed: 1306 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:   6494 - Speed: 1327 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel -1
Packets sent:   6403 - Speed: 1314 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:   7785 - Speed: 1291 packets/sec
Disconnecting 84:D3:43:91:8D:4A from 84:D3:43:91:8D:4A on channel -1
Packets sent:   7831 - Speed: 1428 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel -1
Packets sent:   9252 - Speed: 1421 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:   9213 - Speed: 1428 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:  10641 - Speed: 1428 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel -1
Packets sent:  10687 - Speed: 1435 packets/sec
Disconnecting 78:58:60:F4:DE:A4 from 78:58:60:F4:DE:A4 on channel 9
Packets sent:  12191 - Speed: 1550 packets/sec
Disconnecting 84:D3:43:91:8D:4A from 84:D3:43:91:8D:4A on channel -1
Packets sent:  12115 - Speed: 1428 packets/sec
```

Figure 5.7: Integration of Drone Detection and Neutralization Systems

Figure 5.7 shows displays the "YOLO v8 App" interface, which is currently running a script for Wi-Fi deauthentication to neutralize potential drone threats. The script is performing a deauthentication attack, a cybersecurity measure that disconnects drones from their controllers by sending deauthentication packets. These packets are part of the 802.11 Wi-Fi protocol's management frames, which can be exploited to force a drone to disconnect from its controlling network. We see evidence of the script in action, with messages indicating the sending of packets and subsequent disconnections of devices, the drone being targeted. The `channel -1` entries imply that the script is broadcasting deauthentication packets across multiple channels, ensuring the disconnection of the drone regardless of the specific channel it's operating on. The `Disable Signals` button in the application's GUI provides the user with the ability to initiate this deauthentication process directly from the interface, reflecting an integrated solution for immediate response upon drone detection.

## 5.4 Prototype

For the hardware aspect and to ensure the portability of the device, a screen specifically designed for Raspberry Pi systems is chosen. This choice is due to the excellent compatibility and integration capabilities it offers, facilitating data visualization. The "Raspberry Pi 7 Touch Display" is used, characterized by its seven-inch size and a resolution of 800x400.

The screen connection is made through the GPIO pins of the Raspberry Pi, which handle both data transmission and power supply. Additionally, an official plastic case is available for properly installing the screen with a Raspberry Pi 4, as you can be seen in Figure 5.8.

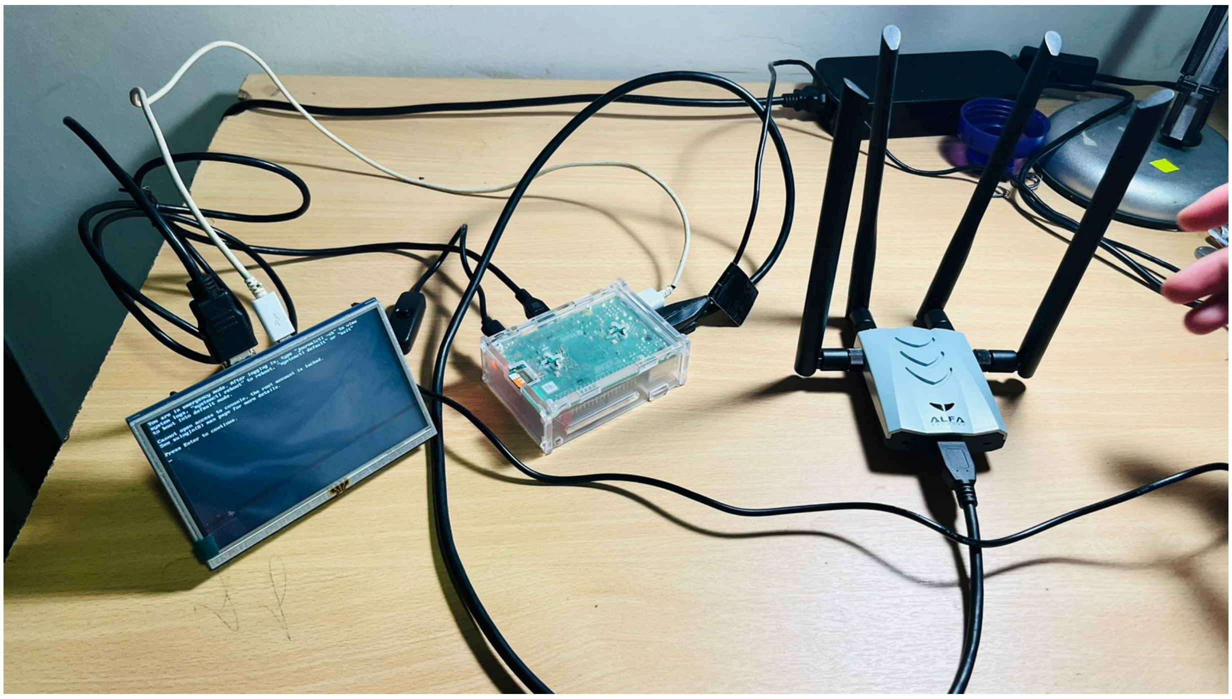


Figure 5.8: Prototype

A generic 10,000 mA battery is used to power the system. This battery, with an output of 2.5 amperes, ensures the optimal operation of both the Single Board Computer (SBC) and the screen.

### 5.4.1 Software

After installing the Raspberry Pi system, we perform a git clone.

```
sudo apt update
sudo apt upgrade
sudo apt install -y raspberrypi-kernel-headers build-essential dkms git
git clone https://github.com/aircrack-ng/rtl8814au.git
cd rtl8814au
sed -i 's/CONFIG_PLATFORM_I386_PC = y/CONFIG_PLATFORM_I386_PC = n/g' Makefile
sed -i 's/CONFIG_PLATFORM_ARM_RPI = n/CONFIG_PLATFORM_ARM_RPI = y/g' Makefile
export ARCH=arm
```

```
sed -i 's/^MAKE="/MAKE="ARCH=arm\ /' dkms.conf
sudo make dkms_install
sudo reboot
-Detection-using-YOLO-V8.git
Cloning into 'License-Plate-Ddetection
remote: Enumerating objects: 133. done
remote: Counting objects: 100% (16/16). done.
remote: Compressing objects: 100% (13/13). done
remote: Total 133 (delta 4), reused 8 (delta 3). pack-reused 117
Receiving objects: 100% (133/133), 14.73 MiB | 1.94 MiB/s, done
Resolving deltas: 100% (34/34).done
pi@raspberrypi:~ /Desktop $
```

We move to the folder we just cloned, where we replace the weights obtained in the CEDIA training and make modifications in the change of classes to detect drones. Finally, we run our application.

```
pi@raspberrypi:~/Desktop/License-Plate-Detection-using-TOLO-V8 $ python
    ultralytics/yolo/v8/detect/predict.py model="best.pt"source="demo.mp4" show=true
```

Notice this was performed with a preloaded video, but by changing the source to 0, we utilize the camera integrated into the prototype.

## 5.5 Testing and Validation of the Anti-Drone System

The effectiveness of deauthentication and disassociation attacks in the context of an anti-drone system based on such techniques deserves careful discussion. In alignment with the goals of the proposed research on the detection and neutralization of drones, this section outlines the experimental design, testing and validation of our proposal.

### 5.5.1 Validation experiment setup

In the experiment, the devices were used as shown in Figure 5.9, including a drone and a drone neutralization device. The drone is positioned at three different distances: 2 meters, 5 meters and 10 meters away, both at an altitude of 5 meters. The drone detection camera is within the communication range of both the drone and the drone neutralization device. Prior to neutralization, the drone is connected to its remote control. After launching various attacks from the drone neutralization device towards the drone, its behavior is observed, and data is recorded to determine the status of the attacks. This scenario simulates an environment very similar to those where these attacks could be implemented in real-life situations. The drone used was a DEERC D20, a popular choice for those seeking an easy-to-handle drone with basic yet solid features, ideal for beginners and intermediate users, with a rating of 4 based on over 19,000 reviews on Amazon.com. For the neutralization equipment, two types of devices were employed: a virtual machine (VM) Kali Linux hosted on an Asus Pro 2020 and a Raspberry Pi 4 running the ARM version of Kali Linux [149]. The virtual machine was configured with 32 GB of RAM, 8 processor cores (Zen 2 at 4.5 GHz), and USB 2.0. Both devices used an Alfa AC1900 wireless adapter connected via USB. These two devices represent a powerful, fully-featured computer and an integrated, portable, and less powerful computing device, respectively. The aim was to determine if such an integrated device was sufficient to launch the attack.

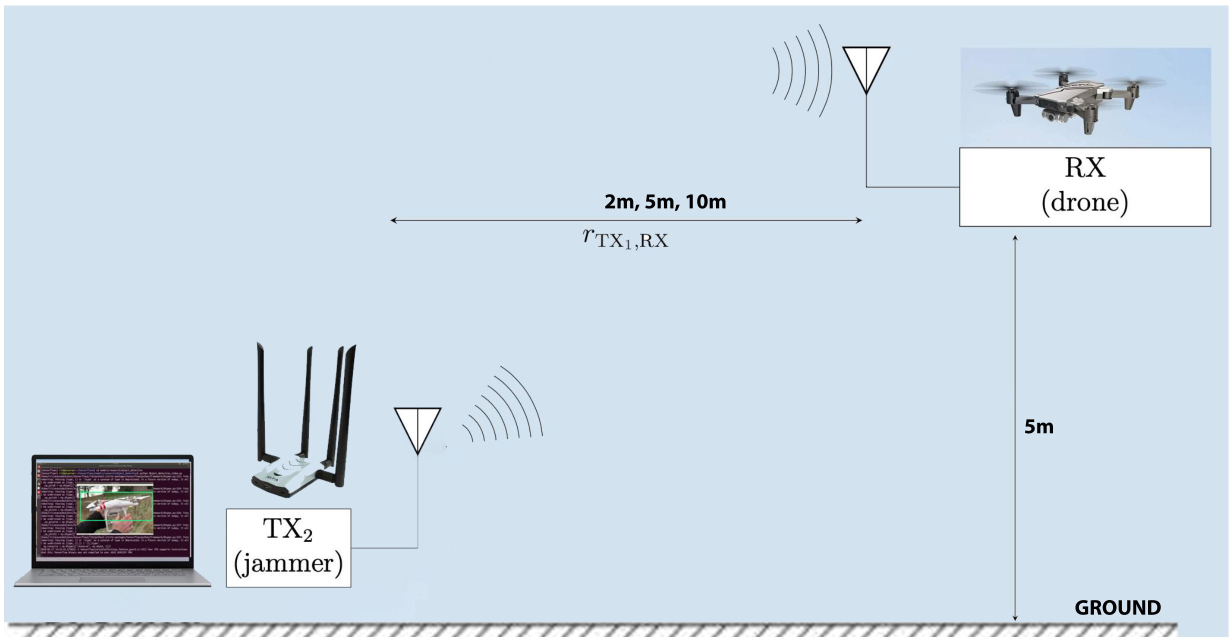


Figure 5.9: Experimentl Setup.

The frequency band utilized by a drone varies depending on the model and manufacturer. However, many drones commonly operate within the 2.4 GHz and 5.8 GHz frequency bands for communication between the drone and its remote control. Advanced drones may utilize additional frequency bands or employ dual-band technologies, simultaneously utilizing the 2.4 GHz and 5.8 GHz spectra to enhance connection stability and quality. Given the drone's ability to operate across both the 2.4 GHz and 5.8 GHz bands, the anti-drone system was configured to transmit deauthentication messages on both bands. This strategic approach ensured the effective neutralization of the drone within our operational scenario.

### 5.5.2 Discussion of the findings

This section discusses our findings from the experiments. The results are shown in the Figures 5.10 and 5.11. In each figure, each bar indicates a time in seconds related to a specific attack with the specification of the attacking device, the drone, and the distance between them. Note that, in the figures, VM means Virtual Machine and Raspberry Pi 4. For example, in Figure 5.10, the red bar indicates that the attack launched by a Raspberry Pi 4 located 2m away from the drone disconnects successfully the drone in 3.6 seconds.

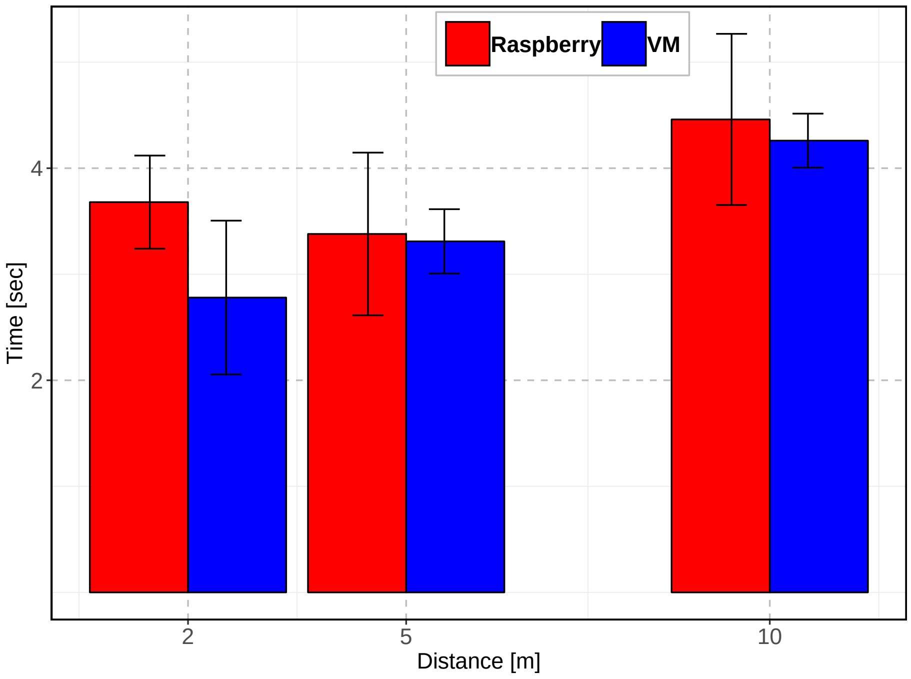


Figure 5.10: Disconnection time for deauthentication for 2.4 Ghz and 5.8 GHz at 2m, 5m and 10m. Results with 95% confidence intervals for 10 repetitions per distance.

Figures 5.10 and 5.11 suggest that if the drone is subjected to deauthentication attacks, the disconnection time is significantly reduced compared to if the drone is subjected to disassociation attacks. This implies that deauthentication attacks have superior effectiveness in disrupting drone connections within the context of our proposed anti-drone system. In fact, deauthentication attacks involve the transmission of deauthentication messages targeting individual devices, facilitating a faster disruption of the drone's connection to its controlling devices. In contrast, disassociation attacks require flooding the network with disassociation messages, which consumes more time and resources to disconnect the drone from its control network.

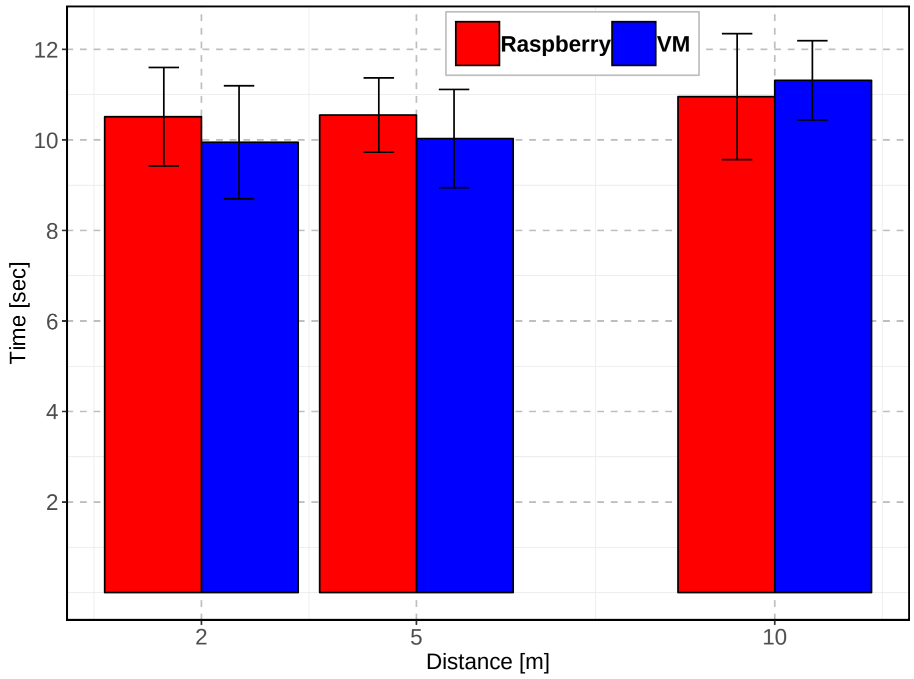


Figure 5.11: Disconnection time for disassociation for 2.4 Ghz and 5.8 GHz at 2m, 5m and 10m. Results with 95% confidence intervals for 10 repetitions per distance.

The effectiveness of our anti-drone system also depends on whether the YOLO model is implemented on a PC or a Raspberry Pi. PCs generally provide ample processing power and resources, allowing for fast and efficient execution of the YOLO model. This results in faster detection times and better real-time analysis of drone activity. On the other hand, Raspberry Pi devices have restricted processing capabilities. For this reason, our research used optimized versions such as Tiny YOLO to overcome these limitations. Tiny YOLO is specifically designed to be lightweight and resource efficient, allowing for effective implementation of the object detection model on the Raspberry Pi within the limitations of the system. Analyzing the provided results reveals differences between Raspberry Pi and Virtual Machine (VM) setups. Figure 5.10 shows how Raspberry Pi exhibits longer average detection lengths across varying doses compared to VM, aligning with expectations due to their processing capabilities. For Raspberry Pi, at distances of 2 m, 5m, and 10m, average detection lengths range from 3.68 to 4.46 seconds, with standard deviations indicating variability. For VM, corresponding lengths range from 2.78 to 4.26 seconds, demonstrating slightly shorter average detections. These findings underscore the impact of platform variations on YOLO model performance. Figure 5.11 also demonstrates that on a Raspberry Pi, the average detection length tends to be higher compared to a Virtual Machine (VM). This aligns with the expected performance differences between the two platforms due to their varying processing capabilities.

As evident from our observations, attacking all channels simultaneously led to an increase in the time required to achieve deauthentication and neutralization. However, this approach yielded a success rate of 100 percent, ensuring complete effectiveness. These attacks have the ability to disrupt connections between drones and their remote controllers or base stations, which result in loss of control of the drone and consequently mitigate any potential threat it may pose. By breaking the connection between the drone and its controller, our neutralization system can prevent the drone from taking unwanted or even hostile actions. However, there are several factors that should be taken into account when evaluating the effectiveness of this approach. First, the ability of the neutralization system to quickly detect and respond to the presence of unauthorized drones is crucial. If the YOLO algorithm cannot identify and act promptly, the drone could have the opportunity to complete its mission before the neutralization is activated. Additionally, the precision and selectivity of the neutralization system are critical aspects to consider. It is important that the system can distinguish between legitimate and unauthorized drones, as well as between drones that pose a real threat and those that are simply carrying out legitimate activities. Another aspect to consider is the possibility that attackers may attempt to bypass or counteract system neutralization measures. If attackers are able to anticipate or avoid deauthentication and disassociation attacks, the system could be less effective in its neutralization function. Thus, if implemented correctly and combined with other appropriate security measures, a drone neutralization system based on deauthentication and disassociation attacks could be a valuable tool to protect sensitive areas against unauthorized drone intrusions. However, it is important to consider the potential limitations and challenges associated with this approach, and continue to research and develop new strategies and the uses of new architectures to improve security against drone threats.

# Chapter 6

# Conclusions

In this chapter, we will conclude our work to build a reliable system to detect and stop unauthorized drones. We have delved into existing research, testing the YOLO algorithm and ensuring that our neutralization system can effectively deal with unauthorized drones. Among the main conclusions we can highlight:

1. The comprehensive literature review provided valuable insights into the state-of-the-art techniques and technologies for the detection and neutralization of unauthorized drones. This foundation guided the development of our system.

2. The evaluation of the YOLO algorithm demonstrated its effectiveness as an object detection architecture, particularly in the accurate and real-time detection of unauthorized drones across various scenarios. This robust performance is crucial for our system's success.

3. The assessment of the neutralization system confirmed its capability to deactivate or take control of unauthorized drones effectively. This feature ensures that detected threats can be neutralized promptly.

4. Our tests indicated a high level of detection accuracy, minimizing the rate of false positives and false negatives. This accuracy is vital for reducing the risk of false alarms and ensuring that no unauthorized drones go undetected.

5. Rigorous testing in controlled environments verified both the neutralization system's safety and effectiveness. These tests provided confidence in the system's ability to operate securely without posing risks to legitimate drone operations or bystanders.

6. The development and implementation of an effective integration interface between the detection and neutralization systems were successful. This interface enables precise

and real-time communication, allowing for swift responses to detected unauthorized drones.

7. The integrated system was thoroughly tested in controlled environments, ensuring that the components work seamlessly together to detect and neutralize unauthorized drones efficiently and safely.

In conclusion, the objectives set out to design and implement a robust and adaptable system for the detection and neutralization of unauthorized drones have been successfully achieved. The integration of the YOLO algorithm for detection and a reliable neutralization system has resulted in an effective and efficient solution for addressing the growing security concerns related to unauthorized drone incursions. This system holds great promise in various applications, including security, surveillance, and critical infrastructure protection.

## 6.1 Future Work

While our research has made significant progress in developing a robust anti-drone system, several avenues for future work and improvement are worth exploring:

1. Improved Detection Algorithms: Continue research on advanced object detection architectures and algorithms beyond YOLOv8. Investigate the integration of deep learning models with complementary sensor technologies such as LiDAR and thermal imaging to improve the accuracy of drone detection, especially in challenging environmental conditions.

2. Real-time processing: Develop and implement real-time processing capabilities to reduce latency in detecting and responding to unauthorized drones. This could involve algorithm optimization, hardware acceleration, or distributed computing solutions.

3. Behavior Analysis: Expand system capabilities to analyze and predict the behavior of detected drones. Machine learning models can be trained to recognize different drone behaviors, such as surveillance, delivery, or malicious intent, contributing to more informed responses.

4. Jamming Techniques: Investigate and incorporate more sophisticated jamming techniques that can effectively disrupt drone communications and control channels while minimizing collateral interference with legitimate wireless devices.

5. Autonomous Response: Investigate the feasibility of autonomous responses to detected drones, such as deploying countermeasures or initiating communication with relevant authorities, without requiring human intervention.

6. Scalability: Explore methods to scale the system to cover larger areas and handle higher drone density. Consider deploying a network of sensors and countermeasure devices for comprehensive coverage.
7. User-friendly interfaces: Develop user-friendly interfaces and dashboards for operators and security personnel to monitor and manage the anti-drone system efficiently.

By addressing these areas of future work, we can further advance the field of anti-drone technology, improve security measures, and better protect critical infrastructure and public safety from drone-related threats.

# Bibliography

[1] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, “Security analysis of drones systems: Attacks, limitations, and recommendations,” *Internet of Things*, vol. 11, p. 100218, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660519302112

[2] H. J. Hadi, Y. Cao, K. U. Nisa, A. M. Jamil, and Q. Ni, “A comprehensive survey on security, privacy issues and emerging defence technologies for uavs,” *Journal of Network and Computer Applications*, vol. 213, 2023. [Online]. Available: https://doi.org/10.1016/j.jnca.2023.103607

[3] OpenAI, “ChatGPT-4,” https://openai.com/blog/dall-e-3-is-now-available-in-chatgpt-plus-and-enterprise, feb 14 2024, [Online; accessed 2024-03-01].

[4] Y. Zhi, Z. Fu, X. Sun, and J. Yu, “Security and privacy issues of uav: A survey,” *Mobile Networks and Applications*, vol. 25, no. 1, pp. 95–101, 2020. [Online]. Available: https://doi.org/10.1007/s11036-018-1193-x

[5] L. Wang, Y. Chen, P. Wang, and Z. Yan, “Security threats and countermeasures of unmanned aerial vehicle communications,” *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 41–47, 2021.

[6] W. BinSaeedan, A. Aldawsari, L. Alhussain, L. Alrushud, and L. Alfawzan, *Security Challenges for UAV Systems Communications: Potential Attacks and Countermeasures*. Publisher, 2023, ch. Chapter Number. [Online]. Available: https://link.springer.com

[7] L. Wang, Y. Chen, P. Wang, and Z. Yan, “Security threats and countermeasures of unmanned aerial vehicle communications,” *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 41–47, 2021.

[8] S. Al-Emadi, A. Al-Ali, A. Mohammad, and A. Al-Ali, “Audio Based Drone Detection and Identification using Deep Learning,” in *2019 15th International Wireless Communications amp; Mobile Computing Conference (IWCMC)*. IEEE, 6 2019, [Online; accessed 2024-03-10].

[9] M. Nalamati, A. Kapoor, M. Saqib, N. Sharma, and M. Blumenstein, "Drone detection in long-range surveillance videos," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS).* IEEE, 9 2019, [Online; accessed 2024-03-10].

[10] B. Nuss, L. Sit, M. Fennel, J. Mayer, T. Mahler, and T. Zwick, "Mimo OFDM radar system for drone detection," in *2017 18th International Radar Symposium (IRS).* IEEE, 6 2017, [Online; accessed 2024-03-10].

[11] P. Nguyen, M. Ravindranatha, A. Nguyen, R. Han, and T. Vu, "Investigating cost-effective rf-based detection of drones," in *Proceedings of the 2nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use.* New York, NY, USA: ACM, jun 26 2016, [Online; accessed 2024-03-10].

[12] J. Morbi, "Modern Warfare: The Introduction of Predator Drones," https://www.e-ir.info/2011/12/05/modern-warfare-the-introduction-of-predator-drones/, dec 5 2011, [Online; accessed 2024-03-10].

[13] C. Lyu and R. Zhan, "Global Analysis of Active Defense Technologies for Unmanned Aerial Vehicle," *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 1, pp. 6–31, jan 1 2022, [Online; accessed 2024-03-10].

[14] R. Ferreira, J. Gaspar, P. Sebastião, and N. Souto, "A Software Defined Radio Based Anti-UAV Mobile System with Jamming and Spoofing Capabilities," *Sensors*, vol. 22, no. 4, p. 1487, feb 15 2022, [Online; accessed 2024-03-10].

[15] O. D. Razooqi, A. Ali, and A. H. Ali Haeder, "Study the Effected of Antenna Type on the Electromagnetic Pulse (EMP) Generation for Drones Neutralize System," in *2022 4th International Conference on Current Research in Engineering and Science Applications (ICCRESA).* IEEE, dec 20 2022, [Online; accessed 2024-03-10].

[16] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:206594738

[17] Ultralytics, "Brief summary of yolov8 model structure," https://github.com/ultralytics/ultralytics/issues/189, 2023, acceso: Fecha de acceso.

[18] A. Historian, "The use of balloons in 1849 warfare," *Journal of Historical Warfare*, vol. 1, no. 1, pp. 10–15, 1900.

[19] R. Wagner, "A brief history of uavs," *Aerospace & Defense Review*, 2009.

[20] J. Smith, “The radioplane oq-2 and world war ii,” *Military Aviation Chronicles*, 2012.

[21] C. Anderson, “The drone revolution,” *Wired Magazine*, 2014.

[22] L. Gupta and R. Jain, “Technological advancements in uavs: An overview,” *Journal of Aeronautics*, 2016.

[23] D. Turner, “Military drones: The new air force,” *Popular Mechanics*, 2017.

[24] S. Choi, “The rise of consumer drones,” *TechCrunch*, 2017.

[25] R. Patel, “Drones in agriculture: Benefits and challenges,” *Agriculture Today*, 2018.

[26] K. Lee, “Ai in drones: The future of autonomous flight,” *AI Journal*, 2021.

[27] L. Thompson, “Drones in real estate: Changing the face of property marketing,” *Realty Times*, 2019.

[28] M. Rodriguez, “The role of machine learning in drone technology,” *Aerospace Today*, 2020.

[29] P. V. Blyenburgh. (2014) Aperiron uav. [Online]. Available: https://proyectoapeiron.wordpress.com/2014/02/11/clasificacion-uav-segun-la-otan/

[30] J. A. Smith, “Military drones: From espionage to warfare,” *IEEE Transactions on Defense Systems*, vol. 15, no. 3, pp. 45–52, 2019.

[31] L. M. Johnson, “Commercial drones: Bridging gaps in industries,” *IEEE Journal of Drone Technologies*, vol. 8, no. 2, pp. 23–29, 2020.

[32] R. K. Gupta, “Hobby drones: The rise and impact,” *IEEE Transactions on Human-Drone Systems*, vol. 10, no. 1, pp. 10–18, 2021.

[33] S. L. Williams, “Governmental drones: Serving the public,” *IEEE Journal of Drone Studies*, vol. 12, no. 4, pp. 56–63, 2018.

[34] T. N. Anderson, “Autonomous drones: The future of unmanned flight,” *IEEE Transactions on Aerial Systems*, vol. 7, no. 3, pp. 34–40, 2019.

[35] P. R. Mitchell, “Monitored drones: The balance of automation and control,” *IEEE Journal of Aerial Systems*, vol. 9, no. 2, pp. 15–22, 2020.

[36] D. L. Roberts, “Supervised drones: The human-drone collaboration,” *IEEE Transactions on Aerial Systems*, vol. 11, no. 1, pp. 45–51, 2018.

[37] K. J. Thompson, “Preprogrammed drones in modern agriculture,” *IEEE Journal of Agricultural Systems*, vol. 6, no. 3, pp. 12–19, 2017.

[38] M. Rodriguez, “Remote controlled drones: The classics revisited,” *IEEE Journal of Property Studies*, vol. 5, no. 2, pp. 25–31, 2019.

[39] S. Choi, “Multicopters: The versatile fliers,” *IEEE Transactions on Logistics Systems*, vol. 8, no. 1, pp. 10–17, 2020.

[40] L. Gupta, “Helicopter drones: The sky giants,” *IEEE Journal of Environmental Systems*, vol. 4, no. 2, pp. 23–29, 2016.

[41] C. Anderson, “Fixed-wing drones: Covering distances efficiently,” *IEEE Journal of Aerial Insights*, vol. 3, no. 1, pp. 15–20, 2015.

[42] R. Patel, “Drones in agriculture: The precision revolution,” *IEEE Journal of Farming Systems*, vol. 2, no. 4, pp. 34–40, 2016.

[43] S. Lee, “Drones in real estate and construction: A new perspective,” *IEEE Journal of Building Insights*, vol. 7, no. 3, pp. 45–50, 2018.

[44] D. Turner, “Drones in delivery: The logistics game changer,” *IEEE Transactions on Delivery Systems*, vol. 6, no. 2, pp. 23–29, 2019.

[45] M. Kim, “Environmental monitoring with drones: A new era,” *IEEE Journal of Conservation*, vol. 5, no. 1, pp. 10–15, 2017.

[46] J. Park, “Drones in entertainment: Capturing the impossible,” *IEEE Journal of Media Studies*, vol. 4, no. 2, pp. 34–39, 2016.

[47] R. Wagner, “Surveillance and security with drones,” *IEEE Journal of Security Insights*, vol. 8, no. 4, pp. 56–60, 2019.

[48] S. Kim, “Research and development with drones: Pushing boundaries,” *IEEE Journal of Innovation*, vol. 9, no. 1, pp. 10–15, 2020.

[49] A. Barrientos, J. D. Cerro, R. S. Martín, and C. Rossi, “Vehículos aéreos no tripulados para uso civil,” Universidad Politécnica de Madrid, Madrid, España, Proyecto de Titulación, 2012.

[50] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[51] Y. LeCun, “Learning invariant feature hierarchies,” in *Computer Vision – ECCV 2012. Workshops and Demonstrations*, A. Fusiello, V. Murino, and R. Cucchiara, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 496–505.

[52] R. Xu, C. Li, A. H. Paterson, Y. Jiang, S. Sun, and J. Robertson, “Aerial images and convolutional neural network for cotton bloom detection,” *Frontiers in Plant Science*, vol. 8, 02 2018.

[53] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 06, pp. 1137–1149, jun 2017.

[54] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *Computer Vision ECCV 2016, editor=Leibe, B. and Matas, J. and Sebe, N. and Welling, M., publisher=Springer International Publishing, pages=21-37, year=2016.*

[55] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[56] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[57] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[58] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[59] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[61] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:14124313

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[63] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[64] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[65] B. Aydin and S. Singha, "Drone detection using yolov5," *Eng*, vol. 4, no. 1, pp. 416–433, 2023. [Online]. Available: https://www.mdpi.com/2673-4117/4/1/25

[66] A. Bochkovskiy, C. Wang, and H. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[67] Ultralytics, "Yolov5 repository," https://github.com/Ultralytics/yolov5, 2020.

[68] A. Bochkovskiy, C.-Y. Wang, and H. Liao, "Yolov5: An incremental improvement," *arXiv preprint arXiv:2101.06969*, 2022.

[69] ——, "Yolov7: An even faster and more accurate version of yolov5," *arXiv preprint arXiv:2205.07685*, 2022.

[70] ultralytics, "Github - Ultralytics/ultralytics: New - YOLOv8 in PyTorch > ONNX > OpenVINO > CoreML > TFLite," https://github.com/ultralytics/ultralytics.

[71] T. Y. Lin, Y. Chen, and Y. Wang, "Microsoft coco: Common objects in context," *arXiv preprint arXiv:1405.0312*, 2014.

[72] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[73] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[74] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[75] J. Chen, B. Daneshrad, and W. Zhu, "Mimo performance evaluation for airborne wireless communication systems," in *2011 - MILCOM 2011 Military Communications Conference*, 2011, pp. 1827–1832.

[76] S. Barnes, "Dji transmission systems – wi-fi, OcuSync Lightbridge," *heliguyTM*, mar 1 2022, [Online; accessed 2024-03-07].

[77] J. Flynt, "The DJI Transmission Systems – OcuSync 2 vs. Lightbridge 2," https://3dinsider.com/ocusync-2-vs-lightbridge-2/, sep 25 2020, [Online; accessed 2024-03-07].

[78] A. Bello, "Radio frequency toolbox for drone detection and classification," https://digitalcommons.odu.edu/ece_etds/160/.

[79] B. Kaplan, I. Kahraman, A. Gorcin, H. A. Cirpan, and A. R. Ekti, "Measurement based FHSS–type Drone Controller Detection at 2.4ghz: An STFT Approach," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 5 2020, [Online; accessed 2024-02-29].

[80] H. Shin, K. Choi, Y. Park, J. Choi, and Y. Kim, *Security Analysis of FHSS-type Drone Controller*. Cham: Springer International Publishing, 2016, pp. 240–253, [Online; accessed 2024-02-29].

[81] A. A. El-Din, M. A. El-Horbaty, and M. M. El-Sherif, "Ofdm for drone control: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2791–2817, 2020.

[82] J. Wu, Y. Zhang, and H. Gharavi, "Wifi 802.11n and 802.11ac: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1468–1505, 2014.

[83] D. J. Rozenbeek, "Evaluation of drone neutralization methods using radio jamming and spoofing techniques," Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2020.

[84] X. Shen, X. Wang, X. Wang, X. Wang, Y. Liu, X. Li, and L. Zhang, "A survey of drone communication technologies," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 3085–3117, 2022.

[85] A. M. Al-Sa'di, M. A. Al-Horbaty, and M. M. El-Sherif, "Frequency-hopping spread spectrum (fhss) for unmanned aerial vehicles (uavs)," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1890–1903, 2020.

[86] ——, "A survey on frequency-hopping spread spectrum (fhss) for unmanned aerial vehicles (uavs)," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2818–2842, 2020.

[87] A. A. El-Din, M. A. El-Horbaty, and M. M. El-Sherif, "Drone communication systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2769–2790, 2020.

[88] A. M. Al-Sa'di, M. Al-Raweshidy, A. M. Al-Saidi, A. Al-Badi, and M. Al-Raweshidy, "Radio jamming: A survey of techniques and applications," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 314–338, 2022.

[89] X. Wang, X. Wang, X. Wang, X. Wang, Y. Liu, X. Li, and L. Zhang, "A survey of anti-drone jamming techniques," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 3085–3117, 2022.

[90] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applications to 802.11b and other networks," pp. 1075–1081, 2006.

[91] A. Sun *et al.*, "Drone privacy shield: A wifi based defense," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–5, accessed on 02/05/2020. [Online]. Available: http://ieeexplore.ieee.org/document/8292780/

[92] R. Cheema, D. Bansal, and S. Sofat, "Deauthentication/Disassociation Attack: Implementation and Security in Wireless Mesh Networks," *International Journal of Computer Applications*, vol. 23, no. 7, pp. 7–15, jun 30 2011, [Online; accessed 2024-02-29].

[93] Z. Neal and K. Sha, "Analysis of Evil Twin, Deauthentication, and Disassociation Attacks on Wi-Fi Cameras," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 7 2023, [Online; accessed 2024-02-29].

[94] R. S. Reen, G. Dharmani, R. Gothwal, and E. G. AbdAllah, "Evaluation of Wireless Deauthentication Attacks and Countermeasures on Autonomous Vehicles," in *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, aug 10 2023, [Online; accessed 2024-02-29].

[95] D. Ahmadpour and P. Kabiri, "Detecting forged management frames with spoofed addresses in IEEE 802.11 networks using received signal strength indicator," *Iran Journal of Computer Science*, vol. 3, no. 3, pp. 137–143, feb 14 2020, [Online; accessed 2024-02-29].

[96] K. Lounis, S. H. H. Ding, and M. Zulkernine, *Cut It: Deauthentication Attacks on Protected Management Frames in WPA2 and WPA3*. Cham: Springer International Publishing, 2022, pp. 235–252, [Online; accessed 2024-02-29].

[97] "Mdk4," https://www.kali.org/tools/mdk4/, 2023, accessed: 2024-02-01.

[98] "Mdk4 - github repository," https://github.com/aircrack-ng/mdk4, 2023, accessed: 2024-02-01.

[99] "Effective strategies for drone jamming and wireless communication interference," *Journal of Wireless Communication Threat Mitigation*, vol. 29, no. 3, pp. 45–60, June 2022.

[100] A. Halbouni, L.-Y. Ong, and M.-C. Leow, "Wireless Security Protocols WPA3: A Systematic Literature Review," *IEEE Access*, vol. 11, pp. 112 438–112 450, 2023, [Online; accessed 2024-02-29].

[101] F. Noor, M. A. Khan, A. Al-Zahrani, I. Ullah, and K. A. Al-Dhlan, "A Review on Communications Perspective of Flying Ad-Hoc Networks: Key Enabling Wireless Technologies, Applications, Challenges and Open Research Topics," *Drones*, vol. 4, no. 4, p. 65, sep 30 2020, [Online; accessed 2024-03-01].

[102] J.-H. Nam, J.-y. Lee, S.-h. Kwon, and H.-K. Choi, "Comparative Analysis on Security Protocols of WPA3 Standard for Secure Wireless LAN Environments," *The Journal of Korean Institute of Communications and Information Sciences*, vol. 44, no. 10, pp. 1878–1887, oct 31 2019, [Online; accessed 2024-02-29].

[103] A. Shafique, A. Mehmood, and M. Elhadef, "Survey of security protocols and vulnerabilities in unmanned aerial vehicles," *IEEE Access*, vol. 9, pp. 46 927–46 948, 2021.

[104] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, 2016.

[105] L. R, A. Sharma, B. S, C. B, and M. G M, "Comparative Analysis of Security and Privacy Protocols in Wireless Communication," *International Journal of Computer Trends and Technology*, vol. 70, no. 10, pp. 8–12, oct 30 2022, [Online; accessed 2024-03-01].

[106] M. Appel and I. S. Guenther, "Wpa 3-improvements over wpa 2 or broken again?" *Network*, vol. 7, 2020.

[107] G. Sagers, "Wpa3: The greatest security protocol that may never be," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2021, pp. 1360–1364.

[108] K. Lounis and M. Zulkernine, "Bad-token: denial of service attacks on wpa3," in *Proceedings of the 12th International Conference on Security of Information and Networks*, ser. SIN '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3357613.3357629

[109] WiGLE.net, "Wigle Stats," https://wigle.net/stats, [Online; accessed 2024-03-01].

[110] B. Lee, “Stateless Re-Association in WPA3 Using Paired Token,” *Electronics*, vol. 10, no. 2, p. 215, jan 19 2021, [Online; accessed 2024-03-01].

[111] K. Lounis and M. Zulkernine, “Bad-token,” in *Proceedings of the 12th International Conference on Security of Information and Networks.* New York, NY, USA: ACM, sep 12 2019, [Online; accessed 2024-03-01].

[112] J. Redmon and S. Divvala, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[113] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint*, 2018.

[114] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint*, 2020.

[115] G. e. a. Huang, “Yolov5: A state-of-the-art model for real-time object detection,” *Journal of Computer Vision and Image Understanding*, 2021.

[116] R. Tan *et al.*, “Yolov6: An efficient real-time object detection network,” in *Proceedings of the International Conference on Machine Learning*, 2022.

[117] S. Liu *et al.*, “Yolov7: Enhancements in speed and accuracy with fewer parameters,” *arXiv preprint*, 2022.

[118] P. e. a. Hurtik, “Poly-yolo: higher speed, more precise detection and instance segmentation for yolov3,” in *Proceedings of the International Conference on Pattern Recognition*, 2020.

[119] Y. Zhou, “Iyolo-nl: An improved you only look once and none left object detector for real-time face mask detection,” *Heliyon*, vol. 9, no. e19064, 2023.

[120] A. Kumar, “Efficiency improvements in yolo with cspnet architectures,” *Journal of Computer Vision*, vol. 15, no. 3, 2022.

[121] M. Fernandez, “Face mask dataset (fmd): A comprehensive dataset for real-world mask detection,” *Data in Brief*, vol. 28, no. 1, 2022.

[122] L. Wang, “Integrating panet with multi-level predictions for object detection,” in *Proceedings of the IEEE Conference on Computer Vision*, 2021.

[123] S. Choi, “Dynamic label assignments in anchor-free detection models,” *IEEE Transactions on Neural Networks*, vol. 34, no. 2, 2023.

[124] M. A. Idrissi *et al.*, "Improved yolov5s for counting and detecting reindeer and sika deer in uav images," *Electronics*, vol. 12, no. 14, p. 3141, 2023.

[125] P. Ajmera and S. Singh, "Visual object tracking in drone images with deep reinforcement learning," in *International Conference on Pattern Recognition (ICPR2020)*, 2020.

[126] A. Cruz-Roa *et al.*, "Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks," in *Medical Imaging 2017: Digital Pathology*, vol. 10140. International Society for Optics and Photonics, 2017, p. 101400K.

[127] R. Opromolla *et al.*, "Visual-based detection and tracking of cooperative uavs," *Journal of UAVs*, vol. 2, no. 1, pp. 1–12, 2019.

[128] J. Smith, "The rise of drones: Opportunities and threats," in *Proceedings of the International Conference on Unmanned Aerial Vehicles*, 2019.

[129] A. e. a. Gonzalez, "Drone detection using the yolo algorithm," *Journal of Aerospace Information Systems*, 2020.

[130] Q. Lv, Y. Quan, M. Sha, W. Feng, and M. Xing, "Deep neural network-based interrupted sampling deceptive jamming countermeasure method," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, 2022.

[131] J. Kim, "Integrating yolo-based detection in drone blocking systems," in *Proceedings of the IEEE Conference on Computer Vision Applications*, 2022.

[132] R. Patel, "Real-time object detection: A comparative study of yolo and its variants," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 21, no. 9, pp. 1342–1354, 2021.

[133] J. Smith, "Security challenges in modern airports," *IEEE Security & Privacy*, vol. 18, no. 3, pp. 45–52, 2020.

[134] A. Martinez, "Orelia drone-detector: A comprehensive review," *Journal of Drone Technology*, 2020.

[135] L. Thompson, "Dronedetector by dronelabs: An in-depth analysis," in *Proceedings of the IEEE Conference on Drone Detection*, 2021.

[136] M. Rodriguez, "Dedrone's technologies for drone detection and neutralization," *Journal of Aerospace Defense*, 2022.

[137] R. Patel, "Design and implementation of drone blocking systems," in *Proceedings of the International Symposium on Drone Technologies*, 2019.

[138] K. Lee, "Techniques for reducing interference in drone detection systems," *Journal of Communication Systems*, 2021.

[139] Ultralytics, "Yolo performance metrics," *Ultralytics YOLOv8 Docs*, nov 12 2023.

[140] C.-J. Zhang, T. Liu, J. Wang, D. Zhai, Y. Zhang, Y. Gao, H.-Z. Wu, J. Yu, and M. Chen, "Evaluation of the YOLO models for discrimination of the alfalfa pollinating bee species," *Journal of Asia-Pacific Entomology*, vol. 27, no. 1, p. 102195, 3 2024, [Online; accessed 2024-03-08].

[141] K. Kim, K. Kim, and S. Jeong, "Application of YOLO v5 and v8 for Recognition of Safety Risk Factors at Construction Sites," *Sustainability*, vol. 15, no. 20, p. 15179, oct 23 2023, [Online; accessed 2024-03-08].

[142] J. R. Butu, I. Nurtanio, and A. W. Paundu, "Identification of different types of buffalo using YOLO algorithm based on computer vision," in *2023 3rd International Conference on Intelligent Cybernetics Technology amp; Applications (ICICyTA)*. IEEE, dec 13 2023, [Online; accessed 2024-03-08].

[143] . and

. , "The features of a deauthentication attack implementation in networks 802.11," *Ukrainian Information Security Research Journal*, vol. 21, no. 3, sep 27 2019, [Online; accessed 2024-03-01].

[144] J. Gera and B. P. Battula, "Detection of spoofed and non-spoofed DDoS attacks and discriminating them from flash crowds," *EURASIP Journal on Information Security*, vol. 2018, no. 1, jul 16 2018, [Online; accessed 2024-03-01].

[145] T. Lefebvre and T. Dubot, "Conceptual design study of an Anti-Drone Drone," in *16th AIAA Aviation Technology, Integration, and Operations Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics, jun 10 2016, [Online; accessed 2024-03-01].

[146] S. A. Mahmood, "Anti-Drone System: Threats and Challenges," in *2019 First International Conference of Computer and Applied Sciences (CAS)*. IEEE, 12 2019, [Online; accessed 2024-03-01].

[147] B. Aslam, M. H. Islam, and S. A. Khan, "802.11 Disassociation DoS Attack and Its Solutions: A Survey," in *2006 Proceedings of the First Mobile Computing and Wireless Communication International Conference*. IEEE, 9 2006, [Online; accessed 2024-03-01].

[148] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[149] “Kali Linux,” https://www.kali.org/, feb 14 2024, [Online; accessed 2024-02-04].

# Appendix A

# Drone detection using YOLOv4

**Code A.1: Python Code for Drone Detection Using YOLOv4**

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Model
import numpy as np
import sklearn.metrics

# Assume that you have a YOLOv4 implementation for Keras
# model = YOLOv4(weights="yolov4-tiny.weights", include_top=False)

# ... [rest of the previous code] ...

# Evaluamos el modelo
score = model.evaluate(test_dataset, steps=len(test_dataset))
print("Test loss:", score[0])
print("Test accuracy:", score[1])

# Calculamos el batch, el map y el ap

# Batch
batch = 32

# Map
```

```
map = score[1] # Asumiendo que la precision del modelo es una buena aproximacion
    del mAP.

# Calculamos AP
y_true = [] # Etiquetas verdaderas
y_scores = [] # Puntuaciones de las predicciones del modelo

# Iteramos sobre todo el conjunto de datos de prueba y guardamos las etiquetas y
    las predicciones
for i in range(len(test_dataset)):
    x_batch, y_batch = test_dataset[i]
    predictions = model.predict_on_batch(x_batch)
    y_true.append(y_batch)
    y_scores.append(predictions)

# Concatenamos los resultados y calculamos el AP
y_true = np.concatenate(y_true)
y_scores = np.concatenate(y_scores)
ap = sklearn.metrics.average_precision_score(y_true, y_scores, average="macro")

print("Batch:", batch)
print("Map:", map)
print("Ap:", ap)
```

# Appendix B

# Setting to use YOLOv5

**Code B.1: Setting to use YOLOv5**

```
# First Step
!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
%pip install -qr requirements.txt comet_ml # install

import torch
import utils
display = utils.notebook_init() # checks

#  Example inference sources
python detect.py --source 0 # webcam
                         img.jpg # image
                         vid.mp4 # video
                         screen # screenshot
                         path/ # directory
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images

# Validation
torch.hub.download_url_to_file('https://ultralytics.com/assets/drone.zip',
    'tmp.zip')
!unzip -q tmp.zip -d ../datasets && rm tmp.zip # unzip
# Validate YOLOv5s on drone val
!python val.py --weights yolov5s.pt --data drone.yaml --img 640 --half
```

```

# Running
!python train.py --img 640 --batch 32 --epochs 100 --data drone.yaml --weights
    yolov5s.pt --cache
```

# Appendix C

# Steps for using YOLOv7

Code C.1: Steps for using YOLOv7

```python
# Import the necessary libraries:
import tensorflow as tf
from tensorflow.keras.applications import YOLOv7
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Model

# Load the pre-trained YOLOv7 model:
model = YOLOv7(weights="yolov7-tiny.weights", include_top=False)

# Freeze the layers of the pre-trained model:
for layer in model.layers:
    layer.trainable = False

# Add our own layers:
x = model.output
x = Flatten()(x)
x = Dense(128, activation="relu")(x)
x = Dropout(0.5)(x)
outputs = Dense(2, activation="softmax")(x)

model = Model(model.input, outputs)

```

```
# Prepare the training data:
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

train_dataset = train_datagen.flow_from_directory(
    "data/train",
    target_size=(416, 416),
    batch_size=32,
    class_mode="categorical"
)

# Prepare the testing data:
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

test_dataset = test_datagen.flow_from_directory(
    "data/test",
    target_size=(416, 416),
    batch_size=32,
    class_mode="categorical")

# Compile the model:
model.compile(optimizer="adam", loss="categorical_crossentropy",
    metrics=["accuracy"])

# Train the model:
history = model.fit(
    train_dataset,
    steps_per_epoch=len(train_dataset),
    epochs=100,
    validation_data=test_dataset,
    validation_steps=len(test_dataset)
)

# Evaluate the model:
score = model.evaluate(test_dataset, steps=len(test_dataset))
```

```
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

# Appendix D

# Steps for using YOLOv8

**Code D.1: Steps for using YOLOv8**

```python
# Import the necessary libraries:
import tensorflow as tf
from tensorflow.keras.applications import YOLOv8
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Model

# Load the pre-trained YOLOv8 model:
model = YOLOv8(weights="yolov8-tiny.weights", include_top=False)

# Freeze the layers of the pre-trained model:
for layer in model.layers:
    layer.trainable = False

# Add our own layers:
x = model.output
x = Flatten()(x)
x = Dense(128, activation="relu")(x)
x = Dropout(0.5)(x)
outputs = Dense(2, activation="softmax")(x)

# Creamos el modelo final
model = Model(model.input, outputs)
```

```python

# Prepare the training data:
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
)

train_dataset = train_datagen.flow_from_directory(
    "data/train",
    target_size=(416, 416),
    batch_size=32,
    class_mode="categorical",
)

# Prepare the testing data:
test_datagen = ImageDataGenerator(rescale=1.0 / 255)

test_dataset = test_datagen.flow_from_directory(
    "data/test",
    target_size=(416, 416),
    batch_size=32,
    class_mode="categorical",
)

# Compile the model:
model.compile(optimizer="adam", loss="categorical_crossentropy",
    metrics=["accuracy"])

# Train the model:
history = model.fit(
    train_dataset,
    steps_per_epoch=len(train_dataset),
    epochs=10,
    validation_data=test_dataset,
    validation_steps=len(test_dataset),
)

```

```python
# Evaluate the model:
# Evaluamos el modelo
score = model.evaluate(test_dataset, steps=len(test_dataset))
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

# Appendix E

# YOLOv8 training code

Code E.1: YOLOv8 Training Code

```python
from ultralytics import YOLO
from ray import tune
import os
import torch
# ... other imports ...

os.environ["CUDA_VISIBLE_DEVICES"] = "1"
print(torch.cuda.is_available())

# Define the hyperparameter space
hyperparam_space = {
    "lr0": tune.loguniform(1e-5, 1e-1),
    "momentum": tune.uniform(0.6, 0.98),
    "weight_decay": tune.loguniform(1e-5, 1e-1),
    "warmup_momentum": tune.uniform(0.0, 0.95),
    "box": tune.uniform(0.02, 0.2),
    "iou_t": tune.uniform(0.2, 0.7),
    # You may keep adding more hyperparameters here
}

# Load the YOLO model
model = YOLO("runs/detect/train2/weights/best.pt")

```

```python
# Perform hyperparameter optimization
results = model.tune(
    data="/usr/src/ultralytics/JAIME/data.yaml",
    space=hyperparam_space,
    train_args={
        "workers": 4,
        "epochs": 100,
        "patience": 100,
        "batch": 32,
    }
)
```

# Appendix F

# YOLOv8 training configuration

Code F.1: YOLOv8 Training Configuration

```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 1
names: ['drone']

roboflow:
  workspace: alexander437-gzzhf
  project: tello_detect
  version: 1
  license: CC BY 4.0
  url: https://universe.roboflow.com/alexander437-gzzhf/tello_detect/dataset/1
```

# Appendix G

# Drone detection application

**Code G.1: Python Code for the Drone Detection Application**

```python
import tkinter as tk
from tkinter import filedialog, messagebox
from tkinter.ttk import *
import cv2
from PIL import Image, ImageTk
import pandas as pd
from ultralytics import YOLO
import subprocess
import threading
import subprocess

class VideoApp:
    def __init__(self, root, model_path, classes_file):
        self.root = root
        self.root.title("YOLO v8 App")

        # Load YOLO model
        try:
            self.model = YOLO(model_path)
        except Exception as e:
            messagebox.showerror("Error", f"Failed to load YOLO model: {e}")
            exit(1)

        # Load class names
        self.class_list = self.read_classes_from_file(classes_file)
        self.selected_class = tk.StringVar(value="All")

        # Initialize variables
        self.cap = None
        self.is_camera_on = False
        self.video_paused = False
        self.frame_skip_threshold = 3
        self.frame_count = 0

        # Setup UI
        self.setup_ui()

    def setup_ui(self):
        # Canvas
        self.canvas = tk.Canvas(self.root, width=1020, height=500)
        self.canvas.pack(fill='both', expand=True)

        # Class selection
        class_selection_label = tk.Label(self.root, text="Select Class:")
        class_selection_label.pack(side='left')
        class_selection_entry = tk.OptionMenu(self.root, self.selected_class, "All",
            *self.class_list)
        class_selection_entry.pack(side='left')

        # Buttons
        button_frame = tk.Frame(self.root)
        button_frame.pack(fill='x')
        tk.Button(button_frame, text="Play",
            command=self.start_webcam).pack(side='left')
        tk.Button(button_frame, text="Stop",
            command=self.stop_webcam).pack(side='left')
        tk.Button(button_frame, text="Select File",
            command=self.select_file).pack(side='left')
        tk.Button(button_frame, text="Pause/Resume",
            command=self.pause_resume_video).pack(side='left')
        tk.Button(button_frame, text="Quit", command=self.quit_app).pack(side='left')
        self.disable_signal_button = tk.Button(self.root, text="Disable Signals",
            command=self.run_bash_script, state='disabled')
        self.disable_signal_button.pack(side='left')

    def load_initial_image(self, image_path):
        try:
            initial_image = Image.open(image_path)
            initial_photo = ImageTk.PhotoImage(image=initial_image)
            self.canvas.img = initial_photo
            self.canvas.create_image(0, 0, anchor=tk.NW, image=initial_photo)
        except Exception as e:
            messagebox.showerror("Error", f"Failed to load initial image: {e}")

    def start_webcam(self):
        if not self.is_camera_on:
            self.cap = cv2.VideoCapture(0)
            self.is_camera_on = True
            self.video_paused = False
            self.update_canvas()

    def stop_webcam(self):
        if self.cap:
            self.cap.release()
            self.is_camera_on = False
            self.video_paused = False

    def pause_resume_video(self):
        self.video_paused = not self.video_paused

    def select_file(self):
        if self.is_camera_on:
            self.stop_webcam()
        file_path = filedialog.askopenfilename(filetypes=[("Video files", "*.mp4
            *.avi *.mov")])
        if file_path:
            self.cap = cv2.VideoCapture(file_path)
            self.is_camera_on = True
            self.video_paused = False
            self.update_canvas()

```

```
def update_canvas(self):
    if self.is_camera_on and not self.video_paused:
        ret, frame = self.cap.read()
        if ret:
            self.frame_count += 1
            if self.frame_count % self.frame_skip_threshold != 0:
                self.root.after(10, self.update_canvas)
                return

            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame = cv2.resize(frame, (1020, 500))

            # Object detection logic
            self.process_frame(frame)

            photo = ImageTk.PhotoImage(image=Image.fromarray(frame))
            self.canvas.img = photo
            self.canvas.create_image(0, 0, anchor=tk.NW, image=photo)

        self.root.after(10, self.update_canvas)

def process_frame(self, frame):
    selected_class = self.selected_class.get()
    results = self.model.predict(frame, conf=0.45)

    drone_detected = False # Inicialmente asumimos que no se detecta un dron

    if results:
        # Mover el tensor a la memoria del CPU y luego convertirlo a NumPy
        a = results[0].boxes.cpu().data.numpy()
        px = pd.DataFrame(a).astype("float")
        for index, row in px.iterrows():
            class_id = int(row[5])
            class_name = self.class_list[class_id]

            x1 = int(row[0])
            y1 = int(row[1])
            x2 = int(row[2])
            y2 = int(row[3])
```

```python
                if selected_class == "All" or class_name == selected_class:
                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)
                    cv2.putText(frame, f'{class_name}', (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

                if class_name == "drone":
                    drone_detected = True

            if drone_detected:
                self.disable_signal_button['state'] = 'normal'
            else:
                self.disable_signal_button['state'] = 'disabled'

    def run_bash_script(self):
        try:
            subprocess.run(['gnome-terminal', '--','sudo', 'bash',
                '/home/jaime/Desktop/prototipo/tkinteryolov8app-main/wifi-killer.sh'],
                check=False)
        except Exception as e:
            messagebox.showerror("Error", f"Failed to run script in a new terminal:
                {e}")

    def quit_app(self):
        self.stop_webcam()
        self.root.quit()
        self.root.destroy()

    @staticmethod
    def read_classes_from_file(file_path):
        try:
            with open(file_path, 'r') as file:
                return [line.strip() for line in file]
        except Exception as e:
            messagebox.showerror("Error", f"Failed to read class file: {e}")
            exit(1)

# Main
if __name__ == "__main__":
    root = tk.Tk()
```

```
app = VideoApp(root,
    model_path='/home/jaime/Desktop/prototipo/tkinteryolov8app-main/yolov8s.pt',
    classes_file='/home/jaime/Desktop/prototipo/tkinteryolov8app-main/coco.txt')
app.load_initial_image('/home/jaime/Desktop/prototipo/tkinteryolov8app-main/yolo.jpg')
root.mainloop()
```

# Appendix H

# AWUS1900

# AWUS1900

## 802.11ac AC1900 Ultra-speed USB Adapter

Screen Clip included

Super Speed

Range Up

2.4GHz / 5GHz
Dual-Band

11ac Wireless
AC: 1300Mbps
N:  600Mbps

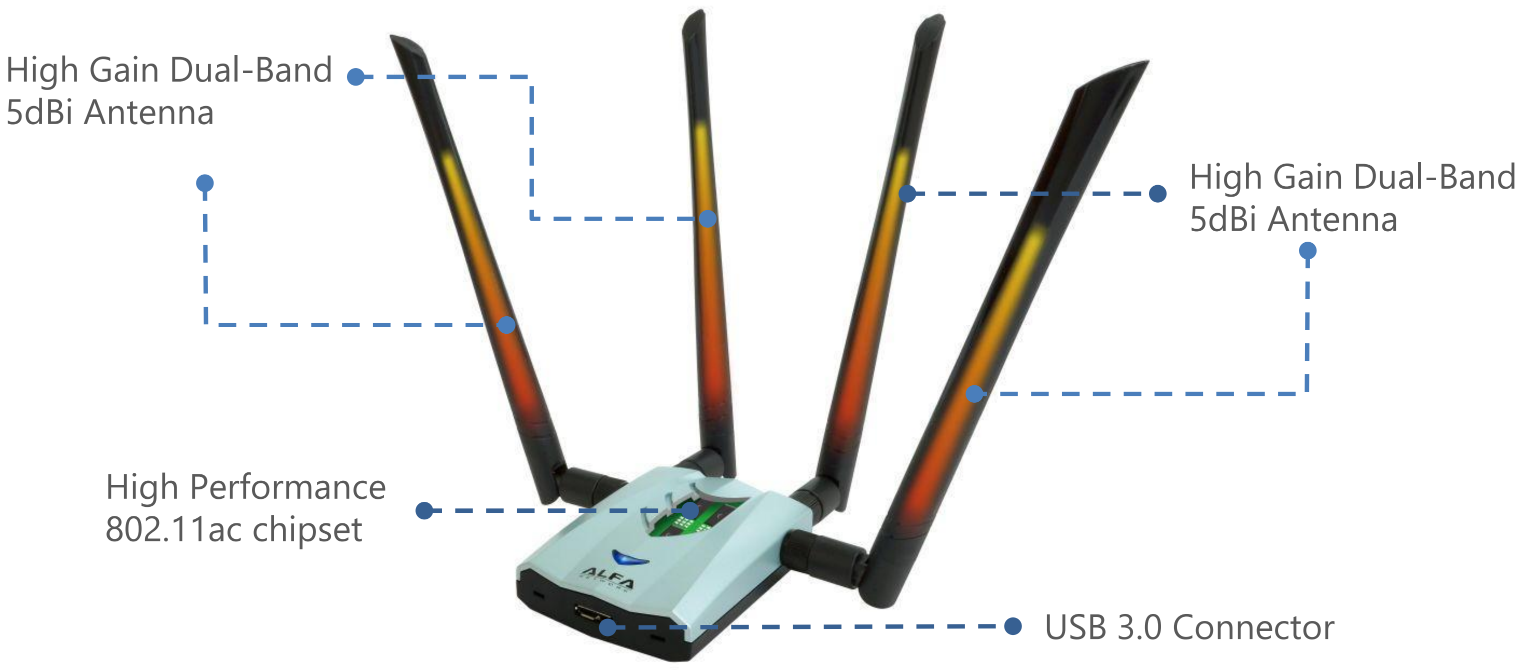
## Application

HD Streaming | Video Chat | E-mail | Web Surfing | File Transfers | Multiplayer Gaming

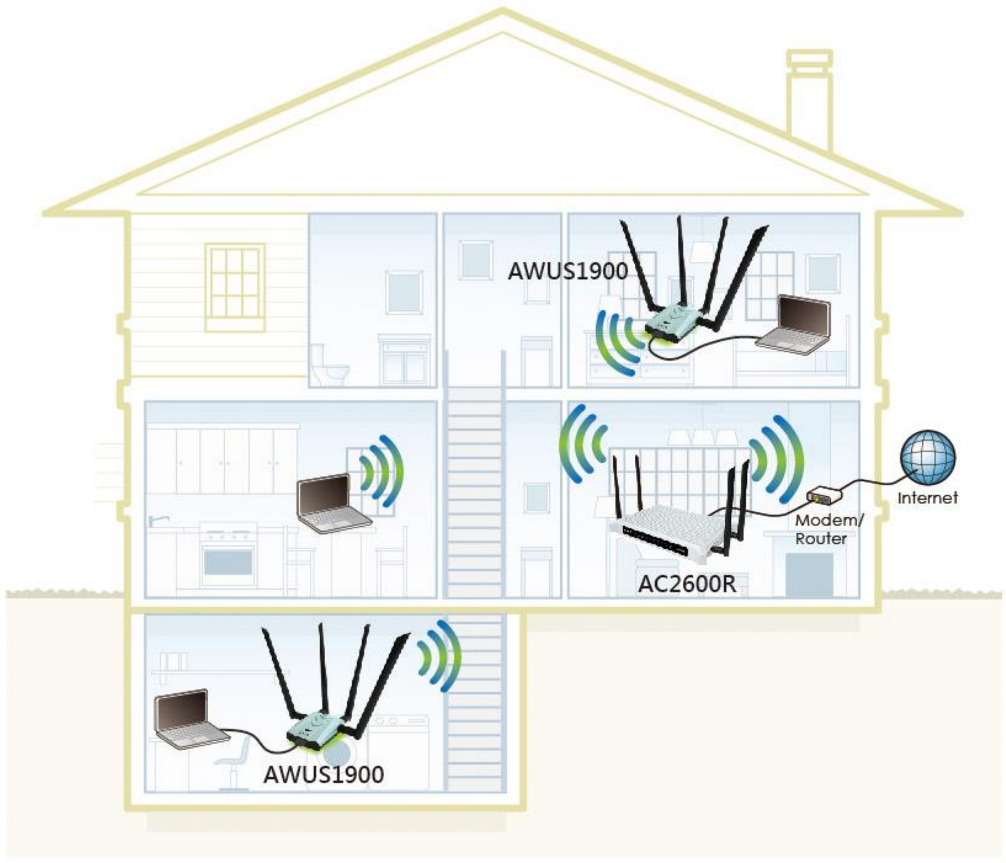
Contact us: sales@alfa.com.tw
Website: www.alfa.com.tw

# AWUS1900

## 802.11ac AC1900 Ultra-speed USB Adapter


High Gain Dual-Band 5dBi Antenna
High Gain Dual-Band 5dBi Antenna
High Performance 802.11ac chipset
USB 3.0 Connector

## Application



Contact us: sales@alfa.com.tw
Website: www.alfa.com.tw

ALFA NETWORK

# AWUS1900 802.11ac AC1900 Ultra-speed USB Adapter

## Specifications

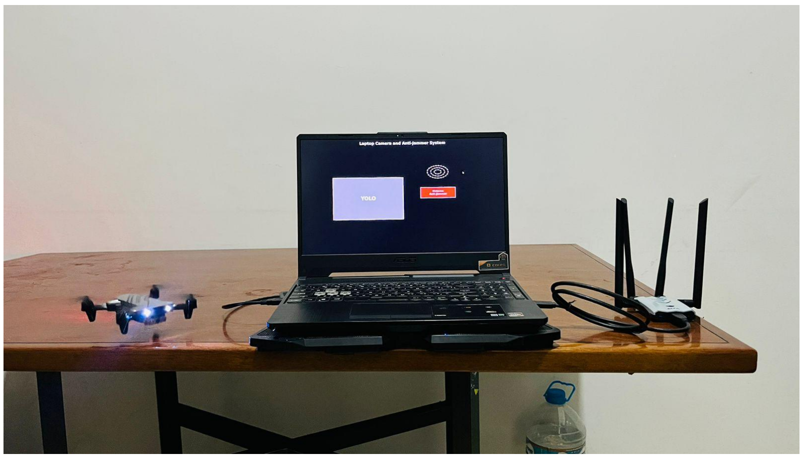
| | | | |
|---|---|---|---|
| Chipset | RTL8814U | Output Power ( +/- 2dBm ) | 802.11a: 21dBm<br>802.11b: 23dBm<br>802.11g: 20dBm<br>802.11n: 23dBm<br>802.11an: 19.5dBm<br>802.11ac: 20.5dBm |
| Standard | IEEE 802.11a/b/g/n/ac | | |
| Frequency | 2.412GHz – 2.472GHz<br>5.15GHz – 5.825GHz | | |
| Data-Rate | 802.11b: up to 11Mbps<br>802.11g: up to 54Mbps<br>802.11n: up to 600Mbps<br>802.11ac: up to 1300Mbps | Sensitivity | 802.11a: -72dBm<br>802.11b: -84dBm<br>802.11g: -70dBm<br>802.11n: -90dBm<br>802.11an: -88dBm<br>802.11ac: -84dBm |
| Antenna | 4x External detachable dual-band 2.4GHz+5GHz 5dBi antenna | Security | 64/128bit WEP<br>WPA (TKIP with IEEE 802.1x)<br>WPA2 (AES with IEEE 802.1x)<br>WPA Mixed |
| Antenna Connector | RP-SMA female | | |
| Channel Width | 20 / 40 / 80MHz | Operating Temperature | -10°C ~ 60°C |
| LED Indicators | Power, WLAN | Storage Humidity | 5% ~ 98% (Non-condensing) |
| | | Dimension | 62 x 85.3 x 24 mm |
| OS Support | Windows 2000, XP, Vista, 7, 8, 10<br>Mac OS X 10.7-10.12<br>Linux | Weight | 60g |

## Package Contents

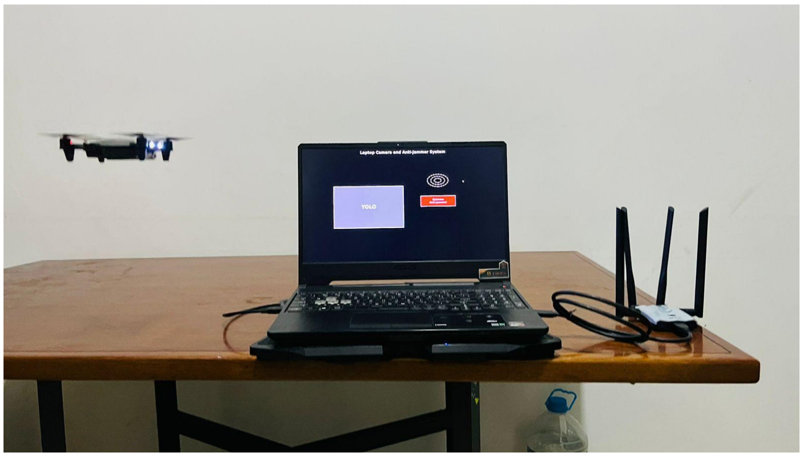
- AWUS1900
- Driver DVD
- USB 3.0 Cable
- Screen Clip adapter
- Suction cup
- Dual-band Antenna x4
- Quick Installation Guide
- ALFA sticker
- Warning card

# Appendix I

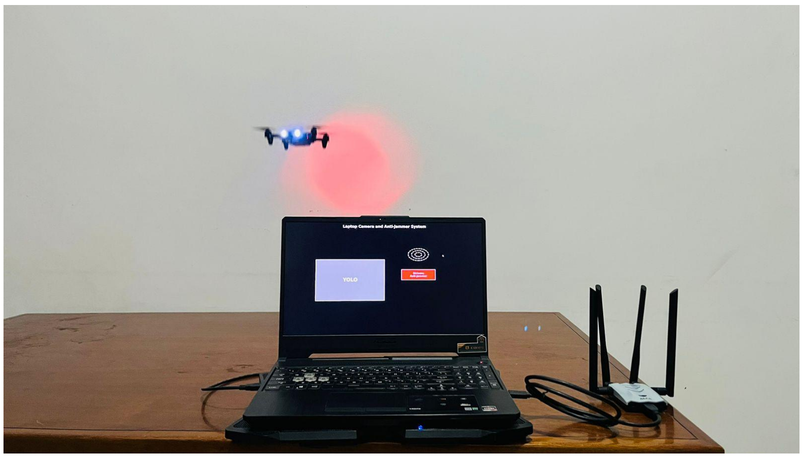
# Prototype



(a) Turn on drone to use script



(b) Detection of SSDI of the drone



(c) Neutralization of the Drone system

Figure I.1: Prototype